webMethods Integration Server 管理者ガイド

バージョン 10.11

2021 年 10 月

WEBMETHODS

このマニュアルは、webMethods Integration Server バージョン 10.11 およびそれ以降のリリースに適用されます。

このマニュアルに含まれる仕様は、変更されることがあります。変更内容については、それ以降のリリースノート、または次のエディションで報告されます。

Copyright © 2007-2021 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

Software AG および、すべての Software AG Software AG, Software AG USA Inc.、またはその子会社やライセンサーの商標または登録商標です。このマニュアルに記載されたその他の企業名および製品名は、それぞれの所有者の商標です。

Software AG またはその子会社、もしくはその両方の商標および特許についての詳細は、次の場所を参照してください。
http://softwareag.com/licenses/

このソフトウェアを使用する際は、Software AG のライセンス契約条件を順守する必要があります。これらの契約条件は、http://softwareag.com/licenses/ またはライセンス製品のルートインストールディレクトリから取得できる製品ドキュメントの一部として参照できます。

本ソフトウェアはサードパーティ製品の一部を含む場合があります。サードパーティの権利、ライセンス条項、追加の権利や制約については、「License Texts, Copyright Notices and Disclaimers of Third Party Products」を参照してください。特定のサードパーティのライセンス条項や制約については、「License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products」の「Legal Notices」の E 項を参照してください。これらのドキュメントは、http://softwareag.com/licenses/ またはライセンス製品のルートインストールディレクトリから取得できる製品ドキュメントの一部として参照できます。

ドキュメントID: IS-AG-1011-20211206_JA

目次

このマニュアルについて **17**
表記規則 18
オンライン情報とサポート 19
データ保護 20

1 管理者の役割 **21**
管理者の役割とは 22
一般的な管理責任 22
Integration Server Administrator とは 23
サーバからの管理メッセージの受信 23
Administrator ユーザ 23
補助管理者の追加 24

2 サーバの概要 **25**
サーバの役割 26
アーキテクチャ 27
サーバによるサービスの実行方法 30
Integration Server のセキュリティ 31
Integration Server ログ 32
Integration Server キャッシング 34
Integration Server の修正について 34

3 サーバの起動および停止 **37**
webMethods Integration Server の起動 38
サーバの起動時に実行される処理 44
Windows アプリケーションまたは Windows サービスとしての Integration Server の実行 45
Mac OS X または他の UNIX システムでファイル記述子を増やす 47
設定ファイル custom_wrapper.conf の設定の変更 47
Integration Server への Java システムプロパティの受け渡し 52
Integration Server のシャットダウン 52
アクティブなセッションの表示 54
セッションの強制終了 55
Integration Server プロセス ID の表示 55
Integration Server の再起動 56
サーバの回復 56
Java サービスラッパー 59

4 複数の Integration Server インスタンスの実行 **63**
Integration Server インスタンスの概要 64
同じマシン上で複数の Integration Server インスタンスを実行する際のガイドライン 64
新規 Integration Server インスタンスの作成について 64
is_instance スクリプトについて 65

新規 Integration Server インスタンスの作成....66
Integration Server インスタンスの更新....70
サーバインスタンスでのパッケージの更新....71
サーバインスタンスのデータベースプロパティの更新....72
サーバインスタンスからパッケージの削除....73
サーバインスタンスでの Language Pack の更新....73
サーバインスタンスの削除....75

5 Integration Server Administrator の使用....77
Integration Server Administrator とは....78
Integration Server Administrator の起動....78
Windows での Integration Server Administrator の起動....79
My webMethods Server から Integration Server Administrator へのアクセス....79
webMethods Integration Server Administrator の概要....80
レガシー Integration Server Administrator へのアクセス....82
新しい Integration Server Administrator における名前とナビゲーションの変更....83
サーバ設定ファイル....84
Integration Server Administrator からのログオフ....84

6 ダッシュボードの使用....85
概要....86
[JVM メトリクス]....88
[使用状況]....90
サービス....92
[API]....97
ダッシュボードの設定....98

7 ユーザとグループの管理....101
ユーザとグループ....102
ユーザアカウントの定義....102
ユーザアカウントの無効化と有効化....115
グループの定義....116

8 サーバの設定....123
ライセンス情報の表示および変更....124
サーバスレッドプールの管理....127
サーバセッションの管理....128
送信 HTTP の設定....131
リモート Integration Server に対するエイリアスの設定....133
送信要求に対するサードパーティのプロキシサーバの指定....137
Integration Server によるログ、状態、その他の情報の保存場所の設定....145
組み込みデータベースから外部 RDBMS への切り替え....146
拡張設定の使い方....146
文字エンコーディングの指定....147
JVM の設定....148
Integration Server の JDK または JRE の指定....148
Integration Server が使用する JVM ヒープサイズの変更....149
Integration Server のアセット情報のパブリッシュおよび取り消し....150

リモートクライアント JMX 監視のポート設定....152
起動中のデバッグ接続を受け入れるためのIntegration Server 設定....153
Integration Server での CORS の使用....154

9 Integration Server を外部 RDBMS のデータベースコンポーネントへ接続する....157
概念....158
接続プールに機能をポイントする....158
MySQL Community Edition データベースドライバのインストール....159
Tibero JDBC ドライバのインストール....160
ドライバエイリアス定義の作成....160
接続プールの作成....160
データベースへの送信接続にケルベロス認証を使用する....165

10 ポートの設定....167
ポートについて....169
ポートの追加に関する考慮事項....172
HTTP ポートの追加....174
HTTPS ポートの追加....180
ファイルポーリングポートについて....188
FTPS ポートの追加....194
FTP ポートの追加....200
電子メールポートの追加....203
HTTP 診断ポートの追加....211
HTTPS 診断ポートの追加....216
HTTP/HTTPS ポートの一時停止....224
HTTP/HTTPS ポートの再開....225
HTTPS 要求のテスト....225
FTP/FTPS ポート範囲の使用....225
プライマリポートについて....227
ポートの削除....228
ポートの編集....228
ポートの有効化/無効化について....228
ポートによるクライアント認証の処理の設定....230
セキュリティプロバイダの追加....230
ポートごとに JSSE 用のプロトコルを無効にする....231
HTTP/S ポートをステートレスに指定する....233

11 サーバログの設定....235
サーバログの概要....236
サーバログに記録する情報の量と種類の指定....237
サーバログエントリをキューに格納するかどうかの指定....239
サーバログのデフォルト場所の変更....239
サーバログをサイズに基づいて循環するように設定する....240
Integration Server によって保持保持されるサーバログファイル数の制限....241
重大な問題に関するメッセージの電子メールアドレスへの送信....241
ログエントリに対する追加処理の実行....243
サーバログの表示....243
ログ表示の変更....244

セッションのログレベルとサーバログの場所の変更....247
グローバリゼーションとサーバログ....248

12 統計ログ....249
統計ログについて....250
統計ログの表示....252
統計ログに影響するサーバ設定パラメータ....253

13 webMethods メッセージングのための Integration Server の設定....255
webMethods メッセージングの概要....256
メッセージング接続エイリアスの使用....256
Universal Messaging Server への接続の認証....281
Broker 接続へのキープアライブモードの指定....283
Integration Server のプライマリポートが変更された場合の Broker クライアントの同期.285
ドキュメントストアの設定....286
Integration Server の非クラスタグループとの負荷分散....294

14 JMS メッセージングのための Integration Server の設定....297
JMS メッセージング設定の概要....298
JNDI プロバイダの使用....298
JMS 接続エイリアスの使用....305
管理オブジェクトの作成....329
接続ファクトリオブジェクトの変更の監視....331
JMS での SSL の使用....336
サポートされている JMS プロバイダ....336
Integration Server クラスパスへの JMS プロバイダクライアントライブラリの追加.....341

15 MQTT メッセージングのための Integration Server の設定....347
Integration Server の MQTT サポートの概要....348
サポートされている MQTT サーバ....349
MQTT 接続エイリアスの作成....350
MQTT 接続エイリアスの編集....354
MQTT 接続エイリアスの有効化および無効化....355
MQTT 接続エイリアスの削除....356
MQTT サーバへの失敗した接続の再試行間隔の指定....356

16 メッセージングの拡張ロギングの使用....357
メッセージングの拡張ロギングについて....358
拡張ロギングのログエントリの内容....358
拡張ロギングにメッセージングログまたはサーバログを使用....361
メッセージングの拡張ロギングの設定....361
メッセージングログの表示....362

17 Universal Messaging クライアントログの設定....363
Universal Messaging クライアントログ....364
Universal Messaging クライアントログの設定....364
イベントライフサイクルログ処理....365

18 Web サービスに対するエンドポイントエイリアスの設定.....369
Web サービスエンドポイントエイリアスについて.....370
HTTP/S で使用するためのプロバイダ Web サービス記述子に対するエンドポイントエイリアスの作成.....371
HTTP/S で使用するためのコンシューマ Web サービス記述子に対するエンドポイントエイリアスの作成.....379
HTTP/S で使用するためのメッセージアドレス指定に対するエンドポイントエイリアスの作成.....391
JMS で使用するためのプロバイダ Web サービス記述子に対するエンドポイントエイリアスの作成.....398
JMS で使用するためのコンシューマ Web サービス記述子に対するエンドポイントエイリアスの作成.....404
JMS で使用するためのメッセージアドレス指定に対するエンドポイントエイリアスの作成.....412
WS セキュリティヘッダーのタイムスタンプ.....418

19 Integration Server での信頼性の高いメッセージ処理の設定.....419
信頼性の高いメッセージ処理の概要.....420
Integration Server での信頼性の高いメッセージ処理の使用.....421
Integration Server での信頼性の高いメッセージ処理の設定.....423
信頼性の高いメッセージ処理のシーケンスレポート.....426
シーケンスを閉じる.....427
シーケンスの終了.....428
確認応答要求の送信.....428

20 JWT を使用するように Integration Server を設定する.....429
JWT の概要.....430
Integration Server での JWT の使用.....431
Integration Server における JWT のサポート.....432
JWT を使用するように Integration Server を設定する.....433

21 ケルベロスを使用するように Integration Server を設定する.....439
Integration Server のケルベロス使用方法の概要.....440
ケルベロスについて.....440
ケルベロス委任認証.....441
ケルベロス設定の前提条件.....443
Integration Server でケルベロス認証を使用するときの制限事項.....443
ケルベロスを使用するように Integration Server を設定する.....444
ケルベロスの JAAS コンテキスト.....447
ケルベロス設定のトラブルシューティング.....447

22 HTTP URL エイリアスの設定.....449
概要.....450
HTTP URL エイリアスの作成.....451
URL エイリアスの部分一致の有効化.....457
HTTP URL エイリアスの表示.....457

URL エイリアスの編集....458
URL エイリアスの削除....459
URL エイリアスの移植性と競合の可能性....459

23 HTTP インターセプタの使用....461
HTTP インターセプタの概要....462
HTTP インターセプタの作成に関する考慮事項....463
HTTP インターセプタの作成....464

24 Integration Server の HTTP 圧縮の設定....467
概要....468
HTTP 要求の圧縮....469
HTTP 応答の圧縮....470

25 Integration Server の SFTP サーバへの接続の設定....473
SFTP の概要....474
SFTP サーバエイリアスの作成....474
SFTP ユーザエイリアスの作成....477

26 Integration Server との通信のセキュリティの設定....483
Integration Server における SSL の概要....484
サポートされている SSL/TLS プロトコル....485
SSL と使用する暗号スイートの指定....488
Integration Server における SSL の設定準備....489
CA の認証の使用: 技術上の考慮事項....494
SSL サーバとしての Integration Server の設定....495
SSL クライアントとしての Integration Server の設定....497
Integration Server JVM の SSL 情報の設定....498
SSL サーバで複数のクライアント認証を使用する....500
WS セキュリティと Integration Server....502

27 SSL セッションログの設定....503
SSL セッションログの概要....504
SSL セッションログの有効化....505
SSL セッションログの表示....505
SSL セッションログのデフォルト場所の変更....505
SSL セッションログのサイズの管理....506
ログの重複エントリを避けるには....506

28 Integration Server におけるキーストアとトラストストアの使用....509
キーストアとトラストストア....510
Integration Server によるキーストアおよびトラストストアの使用方法....511
キーストア、トラストストアおよび鍵のエイリアス....512

29 リソースへのアクセス制御....517
概要....518

リソースへのアクセスをポートごとに制御....519
ディレクティブの使用の制御....532
ACL によるリソースへのアクセス制御....534
ブラックリストへのサービスの追加....545

30 クライアントの認証....549
概要....550
基本認証....550
ダイジェスト認証....551
ケルベロス認証....552
クライアント認証による認証....552
クライアント認証とアクセスコントロール....555
My webMethods のシングルサインオン認証....556
SAML 認証....557

31 JAAS を使用した認証のカスタマイズ....561
概要....562
Integration Server での JAAS の使用....562
JAAS 設定ファイル....562
プラグ接続可能な認証モジュール (PAM)....564
Integration Server 用のカスタム JAAS ログインモジュールの作成....565
JAAS カスタムログインモジュールの例....566

32 マスターパスワードと送信パスワード....571
概要....572
送信パスワードの管理....573
送信パスワードおよびマスターパスワードファイルのバックアップ....573
マスターパスワードの変更....574
マスターパスワードの有効期間の変更....575
configPassman.cnf ファイルについて....575
送信パスワード設定の使用....576
マスターパスワード設定の使用....577
マスターパスワードを紛失または忘れた場合の対処方法....580
起動時にマスターパスワードまたは送信パスワードに問題が生じた場合....580
電子メールリスナーとパッケージの複製....584

33 CSRF ガードによる Integration Server のセキュリティ確保....585
CSRF とは....586
Integration Server で CSRF 攻撃を防止する方法....586
CSRF ガードの用語について....586
Integration Server での CSRF ガードの設定....588
Integration Server で CSRF ガードを設定する場合の制限事項....590

34 webMethods Enterprise Gateway の設定....593
概要....595
Enterprise Gateway の動作....595
Enterprise Gateway Server と内部サーバ間のバージョンの相互運用性....598

従来のサードパーティ製プロキシサーバに対する Enterprise Gateway の利点....600
サービス拒否保護について....600
モバイルアプリケーション保護について....601
モバイルデータの同期について....601
SQL インジェクション対策について....602
アンチウイルススキャンフィルタについて....603
カスタムフィルタについて....604
Enterprise Gateway 設定内のクラスタリング....605
Enterprise Gatewayの設定....605
Enterprise Gateway ポートの設定....607
内部サーバの Enterprise Gateway Server への接続....612
Enterprise Gateway 登録ポートへの接続の表示....616
Enterprise Gateway Serverでのクライアント認証の実行....617
Enterprise Gateway ルールの使用....618
警告オプションの指定....625
サービス拒否攻撃の防止....627
モバイルアプリケーションの使用の制御....630
Enterprise Gateway についてよく寄せられる質問....632

35 OAuth の設定....637
OAuth とは....638
Integration Server での OAuth の使用....639
OAuth クライアントタイプ....640
Integration Server でサポートされる承認の譲渡タイプ....641
Integration Server OAuth サービス....649
OAuth 機能の使用に関する重要な考慮事項....650
OAuth のための Integration Server の設定....651
リソースサーバとしての Integration Server の使用について....670
外部認証サーバの使用....670

36 セントラルユーザディレクトリまたは LDAP の設定....675
操作を開始する前に....676
Integration Server が外部定義されたユーザおよびグループを扱う方法の概要....676
セントラルユーザ管理の設定....678
LDAP の使用の概要....680
LDAP を使用する際のサーバの設定....681
ユーザアカウントとグループに関する考慮事項....688
外部ユーザへの Administrator 特権の付与について....690
外部ユーザへの Developer 特権の付与....691
外部ユーザへのサービスとファイルに対するアクセス特権の付与....692

37 パッケージの管理....695
パッケージの使用....696
サーバによるパッケージ情報の保存....700
パッケージに関する情報の検索....703
パッケージの使用....711
サーバ間でのパッケージのコピー....716
パッケージクラスローダの使用....742

パッケージのホット展開....742
パッケージの自動展開....746

38 サービスの管理....749
サービスとは....750
サービスの完全修飾名....750
サービスとフォルダに関する情報の検索....752
手動によるサーバへのサービスの追加....754
サービスのテスト....754
サービスに関連付けられたスレッドのキャンセルと強制終了....754
パッケージのロード、アンロードまたは複製時に実行するサービス....757
特定のイベントに応答するサービスの実行....758
グローバル変数の管理....759

39 サービスのスケジュール....763
概要....764
ユーザタスクのスケジュール....764
スケジュールされているユーザタスクの表示....769
スケジュールされているユーザタスクの更新....771
ユーザタスクの一時停止....771
一時停止中のユーザタスクの再開....773
スケジュールされているユーザタスクのキャンセル....774
スケジュールされているシステムタスクの表示....774
単純な [繰り返し] オプション....775
[複雑な繰り返し] オプション....776
[ターゲットノード] オプション....779
夏時間に対する移行がスケジュール済みタスクに及ぼす影響....781

40 サービス結果のキャッシング....783
キャッシングとは....784
キャッシュされた結果が返るタイミング....784
サービス結果のキャッシングにパブリックキャッシュを使用する....785
キャッシュのリセット....786
サービスキャッシュの使用状況の監視....787
パブリックキャッシュのサービス結果の表示....788

41 保証付きデリバーの設定....789
保証付きデリバーとは....790
保証付きデリバーのためのサーバ設定....791
保証付きデリバーの管理....795

42 Integration Server での Terracotta Ehcache の設定....799
Terracotta Ehcache とは....800
キャッシュの設定....800
キャッシュとキャッシュマネージャについて....804
キャッシュマネージャの設定ファイル....805
Terracotta ライセンスのインストール、表示および変更....807

オンヒープキャッシュの設定....809
BigMemory キャッシュの設定....811
分散キャッシュの設定....814
キャッシュを検索可能にする....823
キャッシュマネージャの使用....826
キャッシュの使用....835
Terracotta Ehcache アクティビティのログへの記録....849

43 拡張 XML パーサの設定....851
拡張 XML パーサとは....852
拡張 XML パーサの動作の仕組み....852
拡張 XML パーサが使用される状況....854
拡張 XML パーサの設定....855
パーティションサイズの設定....858
ピーク使用率の統計情報の表示....859

44 WebSocket の設定....861
概要....862
WebSocket の動作....862
WebSocket に対する Integration Server の設定....863
WebSocket ポートの設定....863
WebSocketSecure ポートの設定....866
WebSocket ポート用の WebSocket サーバエンドポイントの表示....869
[WebSocket サーバセッション] ページ....870
Integration Server で WebSocket プロトコルを使用する場合の制限....871

45 ロックの管理および最適な実例....873
はじめに....874
ローカルサーバロックまたは VCS 統合ロックの選択....874
ロックの無効化および再有効化....874
最適な実例....876

46 webMethods Messaging Trigger の管理....879
はじめに....880
ドキュメント抽出の管理....880
ドキュメント処理の管理....890
webMethods Messaging Triggerに使用されるサーバスレッド数の制限....898
webMethods Messaging Trigger管理のクラスタ同期....900
webMethods Messaging Triggerプロパティの変更....903
トリガーサービスの再試行およびシャットダウン要求の管理....904
webMethods Messaging Triggerのポーリング要求の遅延....906
Integration Server の以前のバージョンから 10.3 以降への逐次トリガーの移行....909

47 JMS トリガーの管理....911
JMS トリガーの管理の概要....912
JMS トリガーの検索....912
JMS トリガーのステータスと状態について....913

JMS トリガーで使用するスレッドの数の制御....918
Integration Server セッションの再利用の設定....921
JMS セッションの再利用の設定....921
同時実行 JMS トリガーのポーリング要求の遅延....921
JMS トリガーの開始が失敗したときに実行される処理....925
WS エンドポイントトリガーについて....927

48 MQTT トリガー....931
MQTT トリガーの管理の概要....932
MQTT トリガーの状態とステータス....932
MQTT トリガーに対する Integration Server セッションの再利用の設定....934
MQTT トリガーの開始の自動再試行....934

49 重複抑制処理を設定したドキュメント履歴データベースの使用....937
概要....938
ドキュメント履歴データベースからの期限切れエントリの削除....938
重複抑制処理の統計情報の表示....939
重複抑制処理の統計情報の消去....939

50 Integration Server での XA トランザクション管理....941
XA トランザクション管理の概要....942
Integration Server での XA オプションの設定....946
手動でのトランザクションの解決....949

51 Integration Server のコンテンツハンドラ....953
Integration Server がコンテンツハンドラを使用する方法....954
Integration Server が HTTP 要求のためにコンテンツハンドラを選択する方法....954
Integration Server が FTP 要求のためにコンテンツハンドラを選択する方法....954
HTTP および FTP 要求のためのコンテンツハンドラ....954
HTTP 応答のコンテンツハンドラについて....960
FTP 応答のコンテンツハンドラについて....962

52 Integration Server におけるクラスロード....963
Integration Server におけるクラスロードの動作....964
クラスをサーバクラスパスに追加する....968
クラスファイルと jar ファイルの配置場所....974
クラスロードの迅速化....975

53 Integration Server におけるホワイトリストフィルタ処理....977
Integration Server におけるホワイトリストフィルタ処理について....978
ホワイトリストクラスフィルタ処理の有効化または無効化....978
Integration Server ホワイトリストクラスファイル....978
パッケージ用ホワイトリストクラスファイルの作成....979
ホワイトリストにないクラスをログに記録する....980
Integration Server ブラックリストクラスファイル....980

54 保守のためのサーバの休止....983
概要....984
休止設定の指定....987
Integration Server の休止....988
休止モードの終了....991
休止モード設定の展開....991

55 Integration Server の診断....993
はじめに....994
診断ポートの設定....994
診断ユーティリティの使用方法....995
セーフモードでの Integration Server の起動....998
サーバが自動的にセーフモードに移行した場合....999
スレッドダンプの生成....1000

56 エラーログによるサービス例外のデバッグ....1003
はじめに....1004
例外ログの詳細レベルの制御....1004
エラーログの表示....1004
エラーログの解釈....1004

57 Docker での Integration Server の使用....1007
Docker と Integration Server の概要....1008
is_container スクリプトについて....1009
Docker イメージのビルドの前提条件....1010
Microservices Runtime の Docker イメージから Web サービスサポートを除外する....1011
webMethods Cloud でコンシューマに公開するサービスの指定....1012
Integration Server インスタンス用の Docker イメージのビルド....1012
Docker イメージをオンプレミス Docker レジストリにロードする....1018
Docker イメージをオンプレミス Docker レジストリにプッシュする....1019
オンプレミス Docker コンテナでの Docker イメージの実行....1019
Docker イメージを Integration Cloud にプッシュする....1025
Integration Cloud での Docker イメージの実行....1026
Integration Server の Docker コンテナの停止....1027
Docker イメージでの設定変数テンプレートの使用....1027

58 Integration Server Administrator API....1029
Administrator API の概要....1030
認証と承認....1030
REST の URL 構造....1030
CRUD 操作....1031
管理アクション....1032
クレデンシャル....1032
応答の拡張....1033
応答のメディアタイプ....1034
要求ペイロードのメディアタイプ....1034

使用の開始....1035
Administrator API へのアクセスの制御....1035
クライアントエラーのロギング....1036
応答本文にスタックトレースを含める....1036
Integration Server Administrator API 操作....1036

59 JSON 出力の制御....1059
JSON 応答の日付エンコーディングの制御....1060
JSON 出力の書式設定して読みやすくする....1060

60 Command Central を使用した Integration Server の管理....1063
Command Central の概要....1064
Integration Server インスタンス管理....1064
Command Central を使用した Integration Server Administrator へのアクセス....1071
Integration Server インスタンスの監視....1073
Integration Server の設定タイプ....1075
Integration Server のライフサイクルアクション....1082
Integration Server でサポートしているコマンド....1082
UNIX シェルスクリプトを使用した管理製品の接続クレデンシャルの変更....1083
Command Central コマンドラインツールのアップグレード....1084

A Integration Server の展開チェックリスト....1085
はじめに....1086
ステージ 1: インストール....1086
ステージ 2: 基本設定....1087
ステージ 3: ユーザ、グループおよび ACL の設定....1088
ステージ 4: パッケージのパブリッシュ....1089
ステージ 5: 実行時クラスのインストール....1090
ステージ 6: サーバと通信するためのクライアントの準備....1090
ステージ 7: セキュリティの設定....1091
ステージ 8: 起動とテスト....1093
ステージ 9: ソースのアーカイブ....1093

B サーバ設定パラメータ....1095
はじめに....1097
watt.adapters....1097
watt.adminapi....1097
watt.art....1098
watt.broker....1099
watt.brokerCoder....1099
watt.cachedirective....1100
watt.cds....1100
watt.client....1100
watt.config....1102
watt.core....1102
watt.debug....1109
watt.debug2....1111
watt.frag....1112

watt.infradc....1112
watt.net....1112
watt.security....1129
watt.server....1136
watt.serverlog....1276
watt.ssh....1277
watt.ssl....1278
watt.tx....1279
watt.um....1280
watt.wm.tnextdc....1282
watt.wmcloud....1282

C 環境変数....1283
Integration Server と Microservices Runtime に定義された環境変数....1284

D FIPS 140-2 準拠....1291
FIPS 140-2 準拠....1292

E Integration Server がクライアントとして機能するときの NTLM 認証の使用....1293
概要....1294
Java ベースの NTLM サポートの使用....1294
統合 Windows 認証によるネイティブ NTLM サポートの使用....1295

F Integration Server からのユーザデータの削除....1299
ユーザデータの削除....1300
ユーザアカウントへの参照の削除....1300
電子メールアドレスへの参照の削除....1303
ログファイルからの個人データの削除....1304

G Integration Server でのワイヤレス通信....1311
概要....1312
Integration Server とワイヤレスデバイス間の通信方式....1312
Integration Server へのワイヤレス接続のための URL 使用....1313
WML と HDML のサンプル....1316

H Integration Server セッションとサーバログのセッション ID のマスク....1317

I サーバログ機能....1319
サーバログ機能について....1320
Integration Server....1320
WmJSONAPI パッケージ....1326
WmParquet パッケージ....1326
WmXSLT パッケージ....1326
フラットファイル....1327

このマニュアルについて

- 表記規則 18
- オンライン情報とサポート 19
- データ保護 20

このマニュアルは、webMethods Integration Server の管理者を対象としています。このマニュアルでは、サーバの動作の概要について説明します。また、サーバの起動と停止、サーバの設定、ユーザアカウントとセキュリティのセットアップ、パッケージとサービスの管理などの一般的な管理タスクについても説明します。

注:
このマニュアルで説明している機能は、お使いのライセンスバージョンの webMethods Integration Server によって使用できる場合と使用できない場合があります。お使いのシステムでライセンスされているコンポーネントの詳細については、Integration Server Administrator の [**サーバ**] > [ライセンス] ページを参照してください。

Integration Server は webMethods Microservices Runtime で利用可能な機能のサブセットを備えています。お使いの webMethods Integration Server に Microservices Runtime ライセンスが取得されている場合、Microservices Runtime で利用可能な機能を使用できます。Microservices Runtime の詳細については、 *Developing Microservices with webMethods Microservices Runtime* を参照してください。

Integration Server とは異なり、Microservices Runtime では複数のインスタンスを使用できず、OSGi を利用せず、Java Service Wrapper を使用しません。したがって、Integration Server と Microservices Runtime のディレクトリ構造は異なります。Microservices Runtime では Java Service Wrapper を使用して Microservices Runtime にプロパティを渡さないため、一部の設定に違いがあります。

『*webMethods Integration Server Administrator's Guide*』を確認する場合は、Integration Server の *Integration Server_directory* /instances/*instanceName* への参照は Microservices Runtime の *Integration Server_directory* に対応することに注意してください。『*webMethods Integration Server Administrator's Guide*』では、Integration Server と Microservices Runtime に違いがある場合、その点について記載しています。

注:
ディレクトリと設定の違いは Microservices Runtime にのみ適用されます。つまり、Microservices Runtime ライセンスを備えている Integration Server と Microservices Runtime ライセンスを備えていない Integration Server は同じディレクトリ構造を持ち、OSGi を使用し、Java Service Wrapper を利用します。

表記規則

規則	説明
太字	画面上の要素を表します。
縮小フォント	*folder.subfolder.service* 形式のサービス名や場所、API、Java クラス、メソッド、プロパティを表します。
斜体	次の情報を表します。 独自の状況または環境に固有の値を指定する必要がある変数。 本文で最初に出現する新しい用語。 他のドキュメントソースへの参照。

規則	説明
モノスペースフォント	次の情報を表します。 入力する必要があるテキストまたはシステムで表示されるメッセージ。 プログラムコード。
{}	選択肢のセットを表します。ここから 1 つ選択する必要があります。中カッコの内側にある情報のみを入力します。{} 記号は入力しません。
|	構文行で相互排他的な 2 つの選択肢を区切ります。いずれかの選択肢を入力します。| 記号は入力しません。
[ ]	1 つ以上のオプションを表します。大カッコの内側にある情報のみを入力します。[ ] 記号は入力しません。
...	同じ種類の情報を複数回入力できることを示します。情報だけを入力してください。実際のコードに繰り返し記号 (...) を入力しないでください。

オンライン情報とサポート

Software AG マニュアルの Web サイト

マニュアルは、Software AG マニュアルの Web サイト (https://documentation.softwareag.com) で入手できます。

Software AG の Empower 製品サポート Web サイト

Empower のアカウントをまだお持ちでない場合は、氏名、会社名、および会社の電子メールアドレスを empower@softwareag.com まで送信し、アカウントをリクエストしてください。

アカウントを取得したら、Empower (https://empower.softwareag.com/) の [eService] セクションからオンラインでサポートインシデントを開くことができます。

製品情報は、Software AG の Empower 製品サポート Web サイト (https://empower.softwareag.com) で入手できます。

機能および拡張機能に関するリクエストの送信、製品の可用性に関する情報の取得、製品のダウンロードを実行するには、Products に移動します。

修正に関する情報を取得し、早期警告、技術論文、Knowledge Base の記事を読むには、Knowledge Center に移動します。

ご不明な点がありましたら、https://empower.softwareag.com/public_directory.aspx の [Global Support Contact Directory] でお住まいの国のローカルまたはフリーダイヤル番号を検索し、お電話でお問い合わせください。

Software AG Tech Community

マニュアルおよびその他の技術情報は、Software AG Tech Community Web サイト (https://techcommunity.softwareag.com) で入手できます。以下の操作を実行できます。

- Tech Community クレデンシャルを持っている場合は、製品マニュアルにアクセスできます。TECHcommunity クレデンシャルがない場合は、登録し、関心事の領域として [マニュアル] を指定する必要があります。
- 記事、コードサンプル、デモ、チュートリアルにアクセスする
- Software AG の専門家によって承認されたオンライン掲示板フォーラムを使用して、質問したり、ベストプラクティスを話し合ったり、他の顧客が Software AG のテクノロジをどのように使用しているかを学んだりすることが可能です。
- オープンスタンダードや Web テクノロジを取り扱う外部 Web サイトにリンクできます。

データ保護

Software AG 製品は、EU 一般データ保護規則 (GDPR) に従って個人データの処理に関する機能を提供します。該当する場合、それぞれの管理マニュアルに適切な手順が記載されています。

1 管理者の役割

- 管理者の役割とは 22
- 一般的な管理責任 22
- Integration Server Administrator とは 23
- サーバからの管理メッセージの受信 23
- Administrator ユーザ 23
- 補助管理者の追加 24

管理者の役割とは

Integration Server 環境では、管理者が webMethods Integration Server のインストール、設定、および保守を担当します。また、サーバの安全性、クライアントに対する可用性および最大パフォーマンスでの実行を保証することも管理者の責務です。通常、管理者は 1 人ですが、ほとんどのサイトでは少なくともう 1 人のスタッフがその補佐業務を行っています。

一般的な管理責任

webMethods Integration Server の管理者は、次のアクティビティの一部または全部に関与します。

- **サーバのインストールとアップグレード** このアクティビティには、サーバコンピュータに必要なハードウェアとソフトウェアの装備、サーバプログラムのダウンロードとインストール、アップグレードの実装 (必要に応じて行う) などが含まれます。
- **サーバの起動と終了** このアクティビティには、定期的な保守または設定変更などのために必要に応じて行う、サーバのシャットダウンやその後の再起動が含まれます。また、サーバコンピュータのハードウェア/ソフトウェア障害時に行う標準的な回復手順の実行も含まれます。これらのアクティビティの詳細については、37 ページの「サーバの起動および停止」を参照してください。
- **サーバの設定** このアクティビティには、セッションの最大数、ログファイルオプション、ポート割り当てなどの基本操作パラメータの設定が含まれます。これらのアクティビティの詳細については、123 ページの「サーバの設定」を参照してください。
- **ユーザとグループの管理** このアクティビティには、認可ユーザのユーザ名およびパスワードの定義やグループへの認可ユーザの割り当てが含まれます。このタスクの詳細については、101 ページの「ユーザとグループの管理」を参照してください。管理者は、外部システム (LDAP など) からユーザ情報やグループ情報を取得するようサーバを設定することもできます。詳細については、675 ページの「セントラルユーザディレクトリまたは LDAP の設定」を参照してください。
- **サーバのセキュリティ管理** このアクティビティには、他の管理者の識別、各サービスへのアクセスコントロールの割り当ておよびサーバにおけるデジタル認証の使用の設定が含まれます。このタスクの詳細については、561 ページの「JAAS を使用した認証のカスタマイズ」、483 ページの「Integration Server との通信のセキュリティの設定」、561 ページの「JAAS を使用した認証のカスタマイズ」および 483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
- **パッケージとサービスの管理** このアクティビティには、パッケージのアクティブ化/非アクティブ化/コピー、サービスとパッケージの両方またはいずれか一方の更新 (必要に応じて行う) などが含まれます。このタスクの詳細については、695 ページの「パッケージの管理」および 749 ページの「サービスの管理」を参照してください。
- **サーバの複数インスタンスの管理** このアクティビティには、同じマシン上で実行される複数の Integration Server を管理するための前述のすべてまたは一部のアクティビティの実行が含まれます。同じマシン上で複数の Integration Server インスタンスを実行する方法の詳細については、63 ページの「複数の Integration Server インスタンスの実行」を参照してください。

Integration Server Administrator とは

Integration Server Administrator は、管理タスクを実行するために使用するユーティリティです。このユーティリティを使用して、サーバアクティビティの監視、ログ情報の確認、ユーザの追加、サービスの有効化/無効化およびサーバのパフォーマンス機能の調整を行うことができます。Integration Server Administrator の詳細については、77 ページの「Integration Server Administrator の使用」を参照してください。

サーバからの管理メッセージの受信

Integration Server は、さまざまなエラー状態 (内部エラー、バインドエラー、Transaction Manager エラーなど) に関する電子メールメッセージを発行します。エラー発生時にこれらのメッセージを受信したら、管理者として適切な処置を実行してください。

管理者 (または代わりの担当者) がサーバからメッセージを受信するようにするには、794 ページの「エラーメッセージ用の電子メールアドレスと SMTP サーバの指定」の手順に従って、Integration Server Administrator を使用して**電子メール通知**のパラメータを設定する必要があります。

Administrator ユーザ

すべての Integration Server は、Administrator と呼ばれる事前定義済みのユーザアカウントを使用してインストールされます。デフォルト設定では、このユーザだけが Integration Server Administrator で管理タスクを実行できるようになっています。

Administrator のパスワード

Administrator ユーザアカウントに割り当てられている事前定義済みのパスワードは「manage」です。

重要:
このパスワードは、webMethods Integration Server のインストール終了後すぐに変更してください。変更しないと、webMethods がサーバに設定するデフォルトパスワードを知る人物に対して、サーバが無防備な状態になります。Developer ユーザアカウントおよび Replicator ユーザアカウントの事前定義済みパスワードも同様に変更してください。また、Software AG では、事前定義済みアカウントを使用する代わりに各管理者および開発者の個々のアカウントを作成する最良の方法をお勧めします。個々のアカウントの作成後、事前定義済みアカウントを無効にします。

パスワードは、推測しにくいものを指定してください。たとえば、大文字、小文字、数字、特殊文字などを組み合わせて指定します。パスワードは値または空の文字列にできません。名前、電話番号、車のプレートナンバー、保険証番号など、入手しやすい情報は使用しないでください。また、パスワードをメモしないでください。身元のわからない人にパスワードを教えるのも禁物です。

パスワードの変更手順については、106 ページの「Administrator ユーザの追加」を参照してください。

補助管理者の追加

管理者が不在の場合に Integration Server の管理を任せられる補助管理者を少なくとも 1 人以上指定しておくと非常に便利です。

補助管理者をサーバに追加するには、その任に当たるユーザに対して通常のユーザアカウントを作成し (まだ作成していない場合)、そのユーザアカウントを Administrators グループに追加します。

Administrators グループのメンバーのみが Integration Server Administrator を使用できます。ユーザアカウントを作成しそれをグループに追加する場合の詳細については、101 ページの「ユーザとグループの管理」を参照してください。

注:
ユーザ/グループ情報用の外部ディレクトリを使用する場合は、690 ページの「外部ユーザへの Administrator 特権の付与について」の管理者の追加に関する説明を参照してください。

2 サーバの概要

- サーバの役割 26
- アーキテクチャ 27
- サーバによるサービスの実行方法 30
- Integration Server のセキュリティ 31
- Integration Server ログ 32
- Integration Server キャッシング 34
- Integration Server の修正について 34

サーバの役割

webMethods Integration Server は、サービスおよび関連ファイルを含むパッケージをホストします。Integration Server にはコア Integration Server パッケージが同梱されています。たとえば、すべての Integration Server インスタンスには、開発者がサービスやクライアントアプリケーションから呼び出す組み込みサービスや、Integration Server の機能を紹介するサービスが含まれたパッケージがあります。また、別のパッケージを作成して、開発者が作成するサービスを保持することもできます。開発者は、業務システムをパートナの業務システムと統合したり、従来のシステムからデータを抽出したり、データベースをアクセスまたは更新したりする機能を実行するサービスを作成できます。

webMethods Integration Server は、サービスをスムーズで効率的かつセキュアに実行する環境を提供します。また、クライアント要求をデコードして、要求対象サービスを識別し、サービスを呼び出して、適切なフォーマットでデータをサービスに受け渡し、サービスで作成された出力をエンコードしてから、出力をクライアントに返します。

さらにサーバは、クライアントを認証し、クライアントに要求サービスの実行権限があるかどうかを確認し、監査トレールのログを管理して、サービス結果キャッシングなどの機能によりスループットを向上させます。

Integration Server のインスタンスについて

webMethods Integration Server では、1 台のマシン上に複数のサーバインスタンスを作成および実行することができます。webMethods Integration Server をインストールしたら、最初のインスタンスに名前を指定します。最初のインスタンスのデフォルト名は「default」です。Software AG インストーラによって *Software AG_directory* \IntegrationServer\instances ディレクトリにインスタンスが作成されます。Integration Server で提供されるスクリプトや Software AG Command Central を使用して、Integration Server の追加インスタンスを作成できます。

各インスタンスのホームディレクトリは *Software AG_directory* \IntegrationServer\instances\instance にあり、独自のパッケージ、設定ファイル、ログファイルおよび更新が含まれます。各 Integration Server インスタンスに対するパッケージおよび更新は、個別に管理および適用します。Software AG Update Manager を使用して最新の修正を適用できます。

Software AG_directory \IntegrationServer ディレクトリは、作成するすべてのサーバインスタンスの親ディレクトリになります。ここには、すべてのサーバインスタンスが使用する共通ファイルや共有ファイル (共通 jar ファイルや修正など) が含まれます。複数の Integration Server インスタンスを作成および実行する方法については、63 ページの「複数の Integration Server インスタンスの実行」を参照してください。

注:
特に指定のない限り、このガイドにおける「Integration Server」および「サーバ」という用語は、Integration Server インスタンスを指します。

アーキテクチャ

Integration Server は、1 つ以上のポートでクライアントの要求を受信待機します。サーバが各ポートに使用するプロトコルのタイプを指定することができます。サーバでは、HTTP、HTTPS、FTP、FTPS および電子メールのポートがサポートされています。

ほとんどのクライアントは、サーバとの通信に HTTP または HTTPS ポートを使用します。サーバは HTTP と HTTPS を両方サポートしているので、セキュアでないクライアント要求は HTTP ポートで受信待機し、セキュアな要求は HTTPS で受信待機することができます。

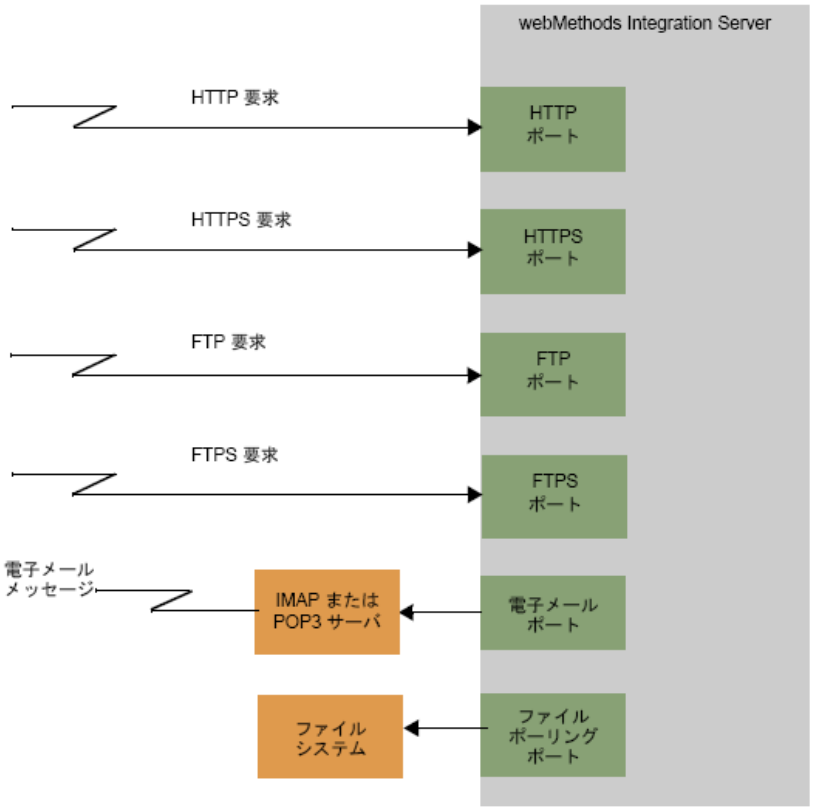
注:
HTTPS と FTPS は、HTTP、FTP および電子メールとは異なり、暗号化および認証によって、セキュアなデータ転送を実現します。HTTPS または FTPS を使用しない場合、未認証ユーザによって、データの取得または変更、IP スプーフィングを使用したサーバへの攻撃、権限を持たないサービスへのアクセス、パスワードの取得などを実行される恐れがあります。パスワードを送信する必要がある場合は、バックエンドのアプリケーションに最低限の特権があることを確認してください。

通常は FTP または FTPS ポートを使用してディレクトリ一覧を取得し、呼び出し対象のサービスを含むディレクトリに切り替え、サービスへの入力を含むファイルを配置して、サービスを実行します。サーバは、サービスが存在するディレクトリにサービス出力を返します。POP3、IMAP などの電子メールサーバ経由で要求を受信するには、電子メールポートを使用します。

各 Integration Server インスタンスには、任意の数のポートを定義できます。デフォルトのサーバインスタンスの HTTP ポートは 5555 です。

注:
また、デフォルトのサーバインスタンスは、9999 に診断ポートを定義します。診断ポートでは HTTP プロトコルが使用され、Integration Server が無応答になった場合にも Integration Server にアクセスすることができます。診断ポートの詳細については、993 ページの「Integration Server の診断 」を参照してください。

指定ポートで要求を受信待機するサーバ



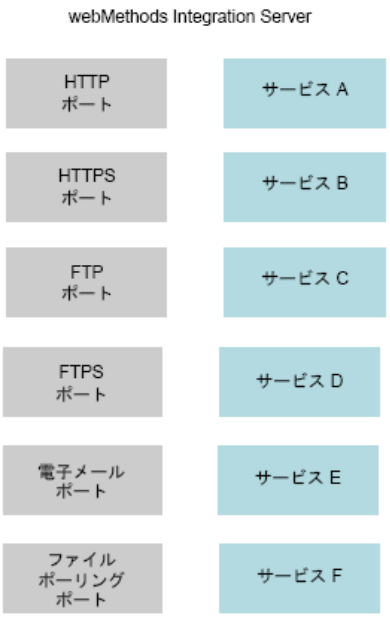
Web サーバが使用する標準ポート番号である、HTTP 要求用のポート 80 や HTTPS 要求用のポート 443 を使用することが望ましい場合があります。Integration Server が Windows システムで稼動している場合は、何の問題もありません。ただし、UNIX システムで Integration Server が稼動している場合、1024 未満のポート番号を使用するにはサーバを root で実行する必要があります。Software AG ではセキュリティ上の理由からこの方法はお勧めしていません。代わりに、権限のないユーザ ID を使用して Integration Server を番号の大きいポート (1024 以上) で実行し、通常ファイアウォールに装備されているポート再マッピング機能を使用して番号の大きいポートに要求を移動します。

サービス

クライアント要求を処理するには、1 つまたは複数のサービスを実行することが必要となります。正常にロードされたサービスは実行可能オブジェクトとして、サーバのプログラムスペース内に保持されます。

サーバを初期化すると、有効パッケージに含まれるサービスがメモリにロードされます。管理者が無効パッケージを有効にすると、そのパッケージのサービスもロードされます。

Integration Server の仮想マシン内でのサービス実行



クライアントがサービスを呼び出すと、サービスは Integration Server プログラム内のスレッドとして動作します。サービスが実現する機能によっては、追加スレッドを作成してタスクを同時実行することもあります。

サービス用データの抽出

サービスが頻繁に実行するタスクには、データソースからのデータ抽出が含まれます。サーバは、ローカルのデータソースから、または Web サーバや JDBC 対応のデータベースなどのリソースに対して HTTP、HTTPS、FTP、FTPS、電子メールまたはファイルポーリング要求を発行することによって、データ (XML や HTML のデータなど) を抽出します。

クライアントから Integration Server にファイルを送信する際には、さまざまな方法を使用できます。Integration Server では、次の自動メカニズムを提供しています。

- HTTP または HTTPS 経由でファイルをサービスにポストする。
- ファイルをサービスに FTP 送信する。
- ファイルポーリングポート経由でファイルをサービスにサブミットする。
- ファイルを電子メールに添付してサービスに送信する。

注:Trading Networks を使用すると、フラットファイルなどの一部の種類のファイルを Trading Networks に直接送信することができます。Trading Networks でフラットファイルを処理する方法の詳細については、『*webMethods Trading Networks Administrator's Guide*』の「Defining and Managing Flat File Document Types」を参照してください。

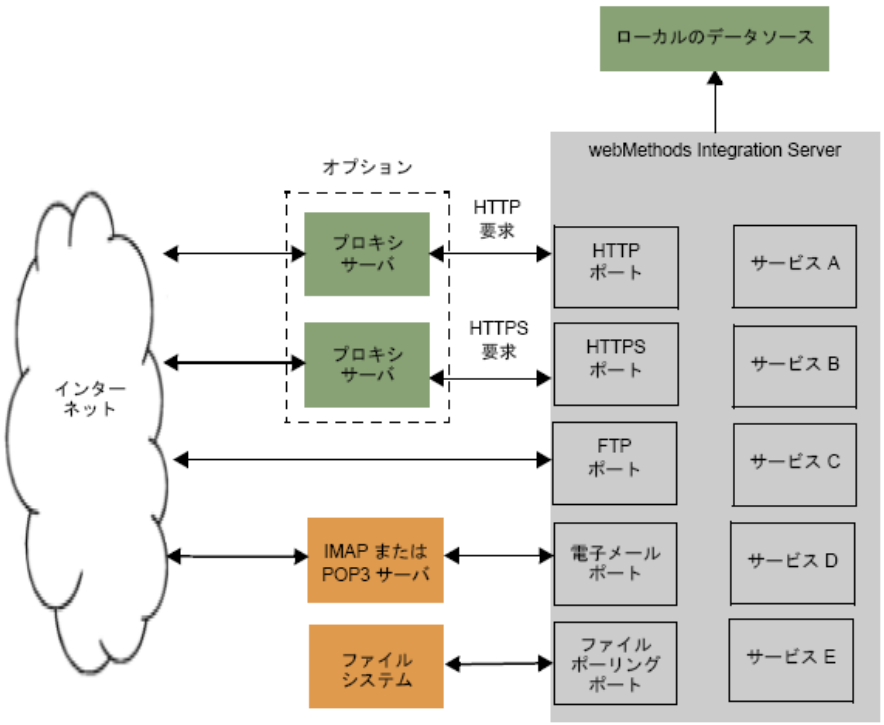
クライアントがファイルを Integration Server にサブミットすると、対応するコンテンツハンドラの実行によってファイルの内容が解析され、ターゲットサービスにファイルが渡されます。

ファイルポーリングを除くすべての伝送方法では、クライアントが指定したサービスが実行されます。ファイルポーリングの場合、そのファイルポーリングポートに関連付けられたサービスが常に実行されます。

XML ファイルの送受信の詳細については、『*webMethods Service Development Help*』を参照してください。フラットファイルの送受信の詳細については、『*Flat File Schema Developer's Guide*』を参照してください。また、自分で作成したサービスから呼び出し可能なサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

サーバが HTTP または HTTPS を使用して外部データソースからのデータにアクセスするときは、要求をプロキシサーバ経由でルーティングすることもできます。

サーバは、ローカルのリソースまたはインターネット上のリソースからデータを取得



サーバによるサービスの実行方法

Integration Server は、クライアントから要求を受信すると、以下の動作を行います。

1. クライアントを認証します。
2. クライアントに対して既にセッションが存在する場合、そのセッションが使用されます。存在しない場合は、セッションが新たに作成されます。

3. 要求されたサービスに合わせてデータを準備できるように、サービス要求のコンテンツタイプを確認します。
4. 入力されたサービス名を使用してサービスを検索します。
5. 要求を受信したポートに基づいて、要求されたサービスへのアクセスが制限されているかどうかを確認します。制限がない場合、サービスの実行が継続されます。
6. サーバは、要求された HTTP メソッドがサービスに対して許可されているかどうかを確認します。許可されていない場合、サーバはエラーメッセージを送信します。許可されている場合、サーバはサービスの実行を継続します。
7. サービスの ACL (Access Control List: アクセスコントロールリスト) を検索し、クライアントにサービスへのアクセス権限が付与されているかどうかを確認します。クライアントがサービスへのアクセスを許可されていることが ACL に記載されている場合は、サーバはサービスの実行を続行します。
8. 監査が有効になっている場合は、監査ログにエントリを追加し、要求の開始を記録します。
9. サービスのサーバ統計情報の収集を開始します。
10. このサービスの結果がサービス結果のキャッシュに存在するかどうかを確認します。サービス結果がキャッシュされていて、キャッシュされた結果の入力がこの要求の入力と一致する場合は、キャッシュされた結果を返します。一致する結果がキャッシュされていない場合は、サービスを呼び出します。サービスが複数のサービスで構成されるフローサービスである場合は、フロー内の各サービスを呼び出します。

    注:
    フロー内のサービスごとに、ステップ 6 から 11 までを実行します。

11. サービスのサーバ統計情報の収集を終了します。
12. 監査が有効になっている場合は、監査ログにエントリを追加し、要求の終了を記録します。
13. コンテンツタイプによって指定された形式でサービス結果をエンコードします。
14. クライアントに結果を返します。

Integration Server のセキュリティ

Integration Server のセキュリティメカニズムによって、不正ユーザによる管理や転送時のデータ傍受を防ぎ、不正アクセスから Integration Server サービスを保護することができます。次のように動作するように Integration Server を設定できます。

- Enterprise Gateway Server を使用して、Integration Server に渡す前に外部クライアントからの要求を途中で取得します。途中で取得することにより、Integration Server を内部ファイアウォールの内側に隔離することができます。
- 接続を認証するために、有効なクレデンシャル (ユーザ名およびパスワード、またはクライアント認証) を提示するようにクライアントに要求します。

- サービスに関連付ける ACL を使用することで、ユーザグループ別に個別のサービスへのアクセスを許可します。最大限のセキュリティを実現するには、すべてのサービスを ACL に関連付けてください。
- Secure Sockets Layer (SSL) を使用した転送レベルのセキュリティと、WS セキュリティを使用したメッセージレベルの Web サービスのセキュリティを提供します。
- ドキュメントにデジタル署名を行って、デジタル署名を検証します。
- サービス要求を受信したポートに基づいて、サービスへのアクセスを制御します。
- Integration Server Administrator にアクセスできるユーザと、Software AG Designer を使用して Integration Server に接続できるユーザを制限します。
- Integration Server Administrator 機能へのアクセスを許可する前に、Administrator 特権を持つ有効なユーザ名およびパスワードを提示するようにクライアントに要求します。
- Integration Server SSL 認証および秘密鍵を業界標準のキーストアファイルに保存することにより、セキュリティ管理を簡素化します。
- それぞれの接続ごとに異なるクライアント認証を使用できるようにします。

Integration Server のセキュリティは、基盤となるオペレーティングシステムのセキュリティにも依存します。必ず次の作業を行ってください。

- 安全に設定するために、ベンダーのすべての推奨事項に従います。
- telnet など、セキュリティの欠陥を含む可能性のある不要なネットワークサービスを削除します。
- セキュリティに影響する可能性のある、オペレーティングシステムベンダーからの更新およびパッチを定期的にチェックして、インストールします。

以上の作業を行う手順については、オペレーティングシステムのマニュアルを参照してください。

セキュリティ監査の詳細については、『*webMethods Audit Logging Guide*』を参照してください。

Integration Server ログ

プラットフォームのログをとることによって、プラットフォームのアクティビティの監視と問題解決に必要となる重要なデータが提供されます。このログデータは Integration Server によって保持されます。ログデータの詳細と使用手順については、『*webMethods Audit Logging Guide*235 ページの「サーバログの設定」』および 363 ページの「Universal Messaging クライアントログの設定」を参照してください。

ログ、データ保護、およびプライバシー

Integration Server のログには、ユーザ名、電子メールアドレス、およびクライアントの IP アドレスなどの個人情報 (PII) が含まれます。監査、サーバアクティビティの監視、および問題の診

断と修正のために、Integration Server のログには個人情報 (PII) が含まれます。Integration Server がログデータを格納する期間はログにより異なります。

ログ	データの格納時間
サーバログ	データはログファイルが存在する間サーバログに残りますが、ログファイルの存在期間は watt.server.serverlogFilesToKeep サーバの設定により異なります。このパラメータは Integration Server がファイルシステムで保持するサーバログファイルの数を制限します。Integration Server がサーバログファイル数の制限に達した場合、Integration Server はサーバログを循環するごとに、アーカイブ済みの最も古いログファイルデータを削除します。 watt.server.serverlogFilesToKeep の詳細については、241 ページの「Integration Server によって保持保持されるサーバログファイル数の制限」を参照してください。 Integration Server サーバログからデータを削除する方法については、1304 ページの「ログファイルからの個人データの削除」を参照してください。
ファイルベースの監査ログ	データはログファイルが存在する間ログに残りますが、ログファイルの存在期間は、watt.server.audit.logFilesToKeep サーバ設定パラメータの値により異なります。このパラメータは、各ロガーに対して Integration Server がファイルシステムに保持する監査ログファイルの数を制限します。監査ロガーのログファイル数が設定された上限に達すると、Integration Server は最も古い監査ログファイルを削除します。 watt.server.serverlogFilesToKeep の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。 Integration Server 監査ログからデータを削除する方法については、1304 ページの「ログファイルからの個人データの削除」を参照してください。
データベースに書き込む監査ログ	データは削除されるまで ISCoreAudit データベース内のログ用のデータベーステーブルに残ります。 Integration Server 監査ログからデータを削除する方法については、1304 ページの「ログファイルからの個人データの削除」を参照してください。
設定変数ログ	データが設定変数ログに残る期間は、オンプレミスの Microservices Runtime のログであるか、Docker コンテナで実行する Microservices Runtime のログであるかによって異なります。設定変数ログはオンプレミスの Microservices Runtime の起動時に上書きされます。Docker コンテナが破棄されると、Docker コンテナで実行する Microservices Runtime の設定変数ログも破棄されます。 設定変数ログからデータを削除する方法については、1309 ページの「設定変数ログからの個人情報の削除」を参照してください。
ラッパーログ	データはログファイルが存在する間ラッパーに残りますが、ログファイルの存在期間は Java サービスラッパーの wrapper.logfile.maxfiles プロパティの値により異なります。Integration Server とともに配信される wrapper.conf file では、このプロパティは 5 に設定されています。このプロパティの詳

ログ	データの格納時間
	細については、60 ページの「ログ記録のプロパティ」を参照してください。 ラッパーログからデータを削除する方法については、1310ページの「Axis2、Kerberos、SAML およびその他のサードパーティライブラリによってログに記録される個人情報の削除」を参照してください。
SSL セッションログ	データは削除されるまで inboundSSLSessions.log に残ります。'watt.net.ssl.server.sessionlog.maxFileSize サーバ設定パラメータは inboundSSLSessions.log ファイルのサイズを制限します。ファイルがサイズの上限に達すると、Integration Server はファイルの名前を sslsession_<DATE(YYYYMMDD)>_TIME(HHMMSS).log に変更し、新しい inboundSSLSessions.log ファイルを作成します。 watt.net.ssl.server.sessionlog.maxFileSize の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。 受信 SSL セッションログからデータを削除する方法については、1308 ページの「SSL セッションログからの個人情報の削除」を参照してください。

Integration Server キャッシング

キャッシングは、頻繁に使用されるデータをメモリに保持することによって、データサービスのパフォーマンスを向上させるための最適化機能です。キャッシングによって、高トラフィックの Web サーバやデータベースから情報を抽出するサービスの応答時間が著しく向上します。

Integration Server は Terracotta Ehcache のキャッシング機能を使用して、以下を提供します。

- **サービス結果のキャッシング**結果をキャッシュするサービスを有効にすると、Integration Server は指定された期間のサービス呼び出し結果をキャッシュに保存します。結果がキャッシュにあると、再度サービスを呼び出すことなく、Integration Server は後続のクライアントのサービス要求に対するサービス結果を迅速に抽出することができます。サービスキャッシュの詳細については、783 ページの「サービス結果のキャッシング」を参照してください。
- **データのキャッシング**Terracotta Ehcache を使用すると、サービスがデータをキャッシュするための個別のキャッシュを定義できます。データのキャッシュを行うサービスは、pub.cache フォルダで提供されるキャッシュサービスを使用してデータをキャッシュします。Terracotta Ehcache を使用すると、事実上すべての種類のオブジェクトをキャッシュできます。また、オフヒープキャッシュや分散キャッシュなどの高度な機能も提供されています。Terracotta Ehcache の使用方法の詳細については、799 ページの「Integration Server での Terracotta Ehcache の設定 」を参照してください。

Integration Server の修正について

Software AG は Software AG Update Manager 経由で Integration Server の修正を提供します。修正には、製品の不具合に対する解決策が含まれるだけでなく、既存の機能の拡張が含まれることがあります。こうした項目は新規または変更された組み込みサービス、API または設定パラメータになることがあります。修正の内容を慎重に確認した後、修正を適用してください。

修正の readme ファイルは修正に提供された機能のマニュアルとして利用できます。通常、リリースされた製品のマニュアルは修正の機能に関して更新されません。リリースされた製品のマニュアルが修正に提供された機能に関して更新される場合、更新された製品マニュアルには機能が利用可能になった修正を示すメモが含まれます。

3 サーバの起動および停止

- webMethods Integration Server の起動 38
- サーバの起動時に実行される処理 44
- Windows アプリケーションまたは Windows サービスとしての Integration Server の実行 45
- Mac OS X または他の UNIX システムでファイル記述子を増やす 47
- 設定ファイル custom_wrapper.conf の設定の変更 47
- Integration Server への Java システムプロパティの受け渡し 52
- Integration Server のシャットダウン 52
- アクティブなセッションの表示 54
- セッションの強制終了 55
- Integration Server プロセス ID の表示 55
- Integration Server の再起動 56
- サーバの回復 56
- Java サービスラッパー 59

webMethods Integration Server の起動

クライアントがサービスを実行するには、webMethods Integration Server が稼動している必要があります。開発環境でサーバを使用している場合、開発者が Software AG Designer を使用してサービスの構築、更新、およびテストを行うためには、Integration Server が稼動している必要があります。

Integration Server を起動する場合は、以下の点に留意してください。

- Integration Server では、一部の設定ファイルおよびログファイルをディスクに保存します。これらのファイルを格納するのに十分な空きディスク容量が Integration Server マシンにあることを確認してください。ディスク容量が不足すると、パフォーマンスに影響したり、エラーが発生したりする可能性があります。
- Integration Server を起動する前に、Integration Server が接続する外部データベースが使用可能であることを確認してください。Integration Server の起動時にデータベースを使用できない場合は、スケジューラなどの一部のコンポーネントが初期化されず、正常に動作しません。

注:
Integration Server のインストール時に、Integration Server Administrator へのログインが成功した直後に Administrator ユーザアカウントのパスワードの変更を要求するオプションを設定できます。設定した場合、サーバが他の処理を行う前に、Administrator ユーザアカウントのパスワードを変更する必要があります。Administrator パスワードを変更する前に Integration Server に送信した要求はすべて拒否されます。

Microservices Runtime の起動の詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

Windows での Integration Server インスタンスの起動および停止

以下の説明に従って、Windows の [スタート] メニューから Integration Server のインスタンスを起動します。

注:
ユーザアカウント制御 (UAC) セキュリティ機能を有効にして Windows Server オペレーティングシステムを実行している場合、Integration Server は完全な Administrator 特権での起動が必要になることがあります。完全な Administrator 特権で Integration Server を起動するには、**Software AG** 用のプログラムまたはアプリケーションのリストに移動します。[*instanceName* の開始] を右クリックし、[管理者として実行] オプションを選択します。オペレーティングシステムに Administrator 特権でログインしていない場合、Administrator クレデンシャルの入力が求められます。

Windows で Integration Server を起動するには

1. [**スタート**] をクリックします。
2. [**すべてのプログラム**] メニューで、 [**Software AG**] フォルダをクリックします。

3. [**サーバの開始**] フォルダをクリックします。

4. [**Integration Server の開始**] をクリックします。

5. [**Integration Server *instanceName* の開始**] をクリックします。

> **注:**
> Integration Server が送信パスワード暗号化のためにマスターパスワードを要求するよう設定されている場合は、このパスワードの入力を促すメッセージがポップアップウィンドウまたはサーバコンソールに表示されます。このパスワードの詳細については、573 ページの「送信パスワードの管理」を参照してください。

UNIX での Integration Server の起動

UNIX で startup.sh を使用して Integration Server を起動する場合は、以下の点に留意してください。

- startup.sh スクリプトを実行するのは、root 以外のユーザ名でログインしたときのみにしてください。root ユーザとしてスクリプトを実行すると、システムのセキュリティが低下する可能性があります。
- UNIX システムの場合、他のシステムよりも多くのファイルやソケットが使用される可能性があります。したがって、Software AG では、UNIX システムでサーバを実行する場合には、少なくとも 1024 個のファイル記述子を使用することをお勧めします。使用可能なファイル記述子を増やすには、サーバの起動前に、UNIX コマンドプロンプトから次のコマンドを入力します。

```
ulimit -n number
```

UNIX で Integration Server を起動するには

1. 次のディレクトリに移動します。

   Software AG_directory \profiles\IS_*instance_name*\bin

   instance_name は Integration Server インスタンスの名前です。

> **注:**
> *Integration Server_directory* \instances*instance_name*\bin ディレクトリに含まれていた startup.bat/sh と shutdown.bat/sh は廃止されています。Software AG では、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリ内のスクリプトの使用をお勧めします。Command Central から Integration Server を管理する場合は、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリにあるスクリプトを使用する必要があります。

2. startup.sh スクリプトファイルを実行します。

> **注:**

Integration Server が送信パスワード暗号化のためにマスターパスワードを要求するよう設定されている場合は、このパスワードの入力を促すメッセージがポップアップウィンドウまたはサーバコンソールに表示されます。このパスワードの詳細については、573 ページの「送信パスワードの管理」を参照してください。

コマンドプロンプトからのサーバインスタンスの起動

コマンドプロンプトからサーバインスタンスを起動できます。サーバインスタンスをコマンドプロンプトから起動するときは、設定ファイルの特定の設定を上書きするオプションがあります。また、サーバインスタンスを「デバッグ」モードで起動して、サーバのアクティビティを記録または表示することもできます。

注:
ユーザアカウント制御 (UAC) セキュリティ機能を有効にして Windows Server オペレーティングシステムを実行している場合、startup.bat サービスの実行に使用するコマンドプロンプトは、完全な Administrator 特権で起動する必要があります。完全な Administrator 特権でコマンドプロンプトを起動するには、[すべてのプログラム] > [アクセサリ] に移動し、[コマンドプロンプト] で右クリックして、[管理者として実行] オプションを選択します。オペレーティングシステムに Administrator 特権でログインしていない場合、Administrator クレデンシャルの入力が求められます。

Integration Server のコマンドプロンプトからサーバインスタンスを起動するには

1. コマンドプロンプトで次のコマンドを入力します。

   ```
   cd Software AG_directory\profiles\IS_instance_name\bin
   ```

   instance_name は Integration Server インスタンスの名前です。

   注:
   Integration Server_directory \instances*instance_name*\bin ディレクトリに含まれていた startup.bat/sh と shutdown.bat/sh は廃止されています。Software AG では、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリ内のスクリプトの使用をお勧めします。Command Central から Integration Server を管理する場合は、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリにあるスクリプトを使用する必要があります。

2. 次のコマンドを入力して、サーバインスタンスを起動します。

Windows の場合:	`bin\startup.bat -switch -switch ...`
UNIX の場合:	`bin/startup.sh -switch-switch ...`

   ここで、*switch* はオプションで、次のいずれかの値を取ることができます。

switch	説明
`-port portNumber`	サーバが HTTP 要求を受信待機するポートを指定します。 *portNumber* は TCP/IP ポート番号を指定します。 例: `-port 8080` このスイッチ変数は、watt.server.port に割り当てられている値を上書きします。 **注:** watt.server.port の値を上書きするだけでなく、`-port` スイッチは、新しい HTTP ポートを WmRoot パッケージに永続的に追加します。この新しいポートは、プライマリポートとして追加され、デフォルト値が含まれます。WmRoot パッケージに同じ TCP/IP 番号のポートが既に存在する場合、`-port` スイッチはその設定を新しいデフォルト値で上書きします。実際には、既存のポートを削除してから新しいポートをデフォルト設定で追加します。 **注:** 80 番ポート (HTTP の標準ポート) や 443 番ポート (HTTPS の標準ポート) を使用する場合、UNIX では、「root ユーザ」として実行する必要があります。セキュリティ上の理由により、上位ポート (たとえば、HTTP 用に 5555 番ポート、HTTPS 用に 5543 番ポート) を使用することをお勧めします。必要に応じてファイアウォールを使用し、80 番ポートを適切なポートに再マッピングしてください。ポートの再マッピングの詳細については、27 ページの「アーキテクチャ」を参照してください。
`-secureport portNumber`	DefaultSecure (HTTPS) ポートのポート番号を指定します。 *portNumber* は TCP/IP ポート番号を指定します。 例: `-secureport 4355` このスイッチ変数は、watt.server.securePort に割り当てられている値を上書きします。 **注:** watt.server.securePort の値を上書きするだけでなく、`-secureport` スイッチは、新しい HTTPS ポートを WmRoot パッケージに永続的に追加します。この新しいポートは、DefaultSecure ポートとして追加され、デフォルト値が含まれます。
`-debug level`	このセッションでのサーバログに記録するデバッグの詳細レベルを指定します。 *level* には、ログに記録する詳細レベルを指定します。

switch	説明

指定する値	記録されるメッセージ
`Fatal`	重大メッセージのみ
`Error`	エラーメッセージおよび重大メッセージ
`Warn`	警告メッセージ、エラーメッセージ、および重大メッセージ
`Info`	情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ
`Debug`	デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ
`Trace`	トレースメッセージ、デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ

現行セッション中、このスイッチ変数は、[**ログ**] > [ログ設定] > [サーバロガーの詳細の表示] ページでデフォルト機能として指定され、watt.debug.level に割り当てられた値を上書きします。

注:
Integration Server 7.1 よりも前の Integration Server では、数値ベースのシステムを使用してサーバログに記録するデバッグ情報のレベルを設定していました。Integration Server では、このシステムとの下位互換性が維持されています。数値ベースのログレベルの詳細については、1095 ページの「サーバ設定パラメータ」の watt.debug.level パラメータの説明を参照してください。

`-log` *`destination`*

このセッションでのサーバログ情報を記録する場所を指定します。*destination* には、次のいずれかを入力します。

オプション	説明
filename	このセッションのサーバログ情報を記録するファイルへの完全修飾パスまたは相対パスを指定します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* \instances*instance_name* に相対的なパスです。 *filename* にはディレクトリとファイル名を指定する必要があります。

switch	説明	
		デフォルトの記録先は watt.debug.logfile サーバ設定パラメータによって制御されます。
	none	サーバログ情報をコンピュータの画面 (STDOUT) に書き込みます。 STDOUTに書き込まれているサーバログメッセージは“ISSERVER”識別子を含み、サーバログメッセージをコンソールに書き込まれた他のメッセージから区別するのに役立ちます。
	both	サーバログ情報をコンピュータ画面 (STDOUT) および watt.debug.logfile パラメータで指定された記録先に書き込みます。

セッション中、*filename* と none の値は watt.debug.logfile に割り当てられた値を上書きします。

注:
SSL セッションログの場合、-log スイッチ値は Integration Server が SSL セッション情報を書き込む先を指定します。-log スイッチの値が none である場合、Integration Server は SSL セッション情報を別のファイル (inboundSSLSessions.log) に書き込みます。スイッチの値が both である場合、Integration Server は SSL セッション情報を inboundSSLSessions.log ファイルおよびコンソールに書き込みます。SSL セッションログの詳細については、503 ページの「SSL セッションログの設定」を参照してください。

注:
none または both の -log スイッチの値は、Microservices Runtime ライセンスを備えている Microservices Runtime や Integration Server がどこに設定変数ログを書き込むかも指定します。ただし、Microservices Runtime はファイル名の値を無視します。ファイル名を指定した場合、Microservices Runtime は設定変数ログファイルを以下の場所だけに書き込みます。
Integration Server_directory /instances/*instanceName*/logs/configurationvariables.log設定変数テンプレートの詳細および関連付けられたログについては、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

-quiesce	サーバを休止モードで起動することを指定します。 休止モードの詳細については、983 ページの「保守のためのサーバの休止」を参照してください。

switch	説明
	注: サーバが休止モードになるときにアセットまたはアクティビティを無効化または一時停止できない場合、警告メッセージが表示されて休止プロセスは続行されます。警告で示された条件が、実行しようとしている保守タスクに干渉する場合は、警告で示された問題を解決してからサーバを休止モードで再起動します。

サーバの起動時に実行される処理

Integration Server を起動すると、サーバでは、クライアントからの要求に備えて、一連の初期化ステップが実行されます。次のプロセスが実行されます。

1. 設定ファイル (*Integration Server_directory* \instances*instance_name*\config\server.cnf) にある設定パラメータを使用して、オペレーティング環境を確立します。
2. 内部管理を実行するプロセスを初期化します。
3. *Integration Server_directory* \instances*instance_name*\packages ディレクトリにあるすべての有効なパッケージとそのサービスに関する情報をロードします。パッケージが別のパッケージに依存している場合、前提条件となるパッケージが先にロードされます。無効なパッケージはロードされません。
4. ロードされたパッケージごとに開始サービスを実行します。
5. 保証付きデリバーエンジンを初期化します。サーバは、ペンディング中の保証付きデリバートランザクション用のジョブストアを確認します。次に、ペンディング中のトランザクションを保証付きデリバーの設定に基づき再試行します。詳細については、789 ページの「保証付きデリバーの設定」を参照してください。
6. 内部システムのタスクのスケジューリングを行います。

サーバの稼動状況を確認する方法

サーバが稼動しているかどうかを確認するには、ブラウザを起動して、Integration Server インスタンスをポイントします(この手順を実行する方法については、78 ページの「Integration Server Administrator の起動」を参照してください)。

- サーバが稼動している場合は、名前とパスワードの入力を促すメッセージが表示されます。
- サーバが稼動していない場合は、ブラウザに次のようなエラーメッセージが表示されます。

```
「このページは表示できません。」
「サーバへの接続を確立できませんでした。」
```

起動時にマスターパスワードや送信パスワードに問題があることが Integration Server で検知されると、セーフモードになります。これは、問題の診断と修正を行うことができる特別なモードです。Integration Server がセーフモードになると Integration Server Administrator が表示されますが、Integration Server は外部リソースには接続されません。

マスターパスワードまたは送信パスワードに問題があるためにセーフモードになっている場合は、Integration Server Administrator の [サーバの統計情報] ページの左上隅に次のメッセージが表示されます。

```
サーバがセーフモードで実行中です。マスターパスワードの健全性チェックが失敗しました。無効な
マスターパスワードが提供されました。
```

これらの問題の原因は、マスターパスワードファイルが破損しているか、送信パスワードファイルが破損しているか、マスターパスワードの入力を求められたときの単純なタイプミスである可能性があります。誤ったパスワードを入力したと思われる場合は、サーバをシャットダウンして再起動し、正しいパスワードを入力します。以上の操作によって問題が解決されない場合は、580 ページの「起動時にマスターパスワードまたは送信パスワードに問題が生じた場合」を参照してください。

Windows アプリケーションまたは Windows サービスとしての Integration Server の実行

Integration Server は Windows アプリケーションまたは Windows サービスとして実行できます。

- **Windows アプリケーションを使用** Integration Server の初期化を制御します。Integration Server が Windows アプリケーションである場合は手動で起動する必要があります。

  Integration Server を Windows サービスとしてインストールしているが Windows アプリケーションとして実行したい場合は、Integration Server の Windows サービスを手動で削除できます。Windows サービスを削除した後でも、Integration Server は Windows アプリケーションとして使用できます。46 ページの「Windows サービスから Windows アプリケーションへのサーバの切り替え」を参照してください。

- **Windows サービスを使用** Integration Server がインストールされたマシンの初期化時に自動的に初期化されるようにします。Windows サービスを使用すると、マシンの再起動後に Integration Server を手動で再起動する必要がありません。

  Integration Server を Windows アプリケーションとしてインストールしているが Windows サービスとして実行したい場合は、Integration Server サービスを手動で登録できます。

  46 ページの「Windows アプリケーションから Windows サービスへのサーバの切り替え」を参照してください。

  注:
  サーバ初期化時に送信パスワードのマスターパスワードを入力するように促すメッセージを Integration Server から出力する場合は、Windows サービスとして実行しないでください。送信パスワードとマスターパスワードの詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。

1 台のコンピュータ上で、複数の Integration Server インスタンスをアプリケーションとして、複数の Integration Server インスタンスをサービスとして実行できます。たとえば、Integration Server のインスタンス 2 つをアプリケーションとして実行し、同じマシンで Integration Server のインスタンス 2 つをサービスとして実行できます。

注:

Microservices Runtime は Windows サービスとしては登録できません。Microservices Runtime ライセンスを持つ Integration Server は Windows サービスとして登録できます。

Windows サービスから Windows アプリケーションへのサーバの切り替え

Integration Server を Windows サービスとしてインストールしていて、Integration Server を Windows アプリケーションとして実行したい場合は、Integration Server の Windows サービスを削除する必要があります。

Integration Server の Windows サービスを手動で削除するには

1. Windows サービスが実行中であれば、停止します。Windows サービスを停止するには、Integration Server Administrator で Integration Server をシャットダウンするか、Microsoft Windows コントロールパネルの [**サービス**] ダイアログボックスから行います。

2. コマンドプロンプトを開いて、 *Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリに移動し、次のコマンドを実行して Integration Server サービスを作成します。

   ```
   service.bat -remove
   ```

   注:
   Integration Server_directory \instances*instance_name* \support\win32 ディレクトリ内の installSvc.bat ファイルは廃止されました。Software AG では、 *Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリの service.bat ファイルを使用することをお勧めします。

Windows アプリケーションから Windows サービスへのサーバの切り替え

サーバを Windows アプリケーションから Windows サービスに切り替えるには、Integration Server インスタンスを手動で登録して Windows サービスとして実行する必要があります。

注:
Integration Server を Windows サービスとして実行するために使用されるユーザは Windows Administrator であるか Windows Administrators グループのメンバーである必要があります。

Integration Server インスタンスを Windows サービスとして実行するように手動で登録するには

1. custom_wrapper.conf ファイルを環境に合わせて編集します。

   たとえば、`wrapper.java.initmemory` プロパティを更新して、Java の最小ヒープサイズを変更できます。custom_wrapper.conf ファイルは、 *Software AG_directory* \profiles\IS_*instance_name*\configuration ディレクトリにあります。

custom_wrapper.conf ファイル内での値の設定の詳細については、『*Software AG Infrastructure Administrator's Guide*』を参照してください。

2. コマンドプロンプトを開いて、 *Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリに移動し、次のコマンドを実行して Integration Server サービスを作成します。

```
service.bat -install
```

> **注:**
> *Integration Server_directory* \instances*instance_name* \support\win32 ディレクトリ内の installSvc.bat ファイルは廃止されました。Software AG では、 *Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリの service.bat ファイルを使用することをお勧めします。

Microsoft Windows コントロールパネルの [サービス] ダイアログボックスで、Integration Server のサービスが作成されたことを確認します。

3. 次のいずれかの方法でサービスを開始します。

- Microsoft Windows コントロールパネルの [**サービス**] ダイアログボックス
- 次のコマンドを使用して、コマンドプロンプトに入力

```
net start <SVCNAME>
```

<SVCNAME> は、前の手順で作成したサービスの名前です。

Mac OS X または他の UNIX システムでファイル記述子を増やす

Integration Server でトラフィックを処理する能力は、Integration Server プロセスで使用できるファイル記述子の数によって制限されます。ほとんどのシステムでは、プロセスごとにデフォルトで 64 個のファイル記述子を使用できます。Integration Server が Mac OS X または他の UNIX システム上で実行されている場合、Software AG では、Integration Server プロセスで使用できるファイル記述子の数を 1024 以上に増やすことをお勧めします。

> **注:**
> Integration Server で同時に開く必要があるファイルの数によっては、この数の増加が必要なことがあります。select 関数の制限により、rlim_fd_max 値を 1024 より大きな値に設定すると危険なため、Integration Server でさらにファイル記述子が必要な場合、setrlimit 値を直接設定してください。

設定ファイル custom_wrapper.conf の設定の変更

Integration Server の起動時に使用される custom_wrapper.conf には、設定が含まれます。Integration Server のシングルセッションの設定を変更するには、コマンドプロンプトから Integration Server を起動し、スイッチを使用して custom_wrapper.conf ファイルの値を変更します。ただし、場合によっては、変更された設定値をすべてのセッションで使用するよう、custom_wrapper.conf の設定値の一部に対して永続的な変更が必要なことがあります。

> **注:**

Microservices Runtime では custom_wrapper.conf を使用しません。Microservices Runtime の場合、 *Integration Server_directory* /bin/setenv.bat(sh) ファイルで設定情報を設定します。

設定ファイルの設定を変更するには

1. テキストエディタで、次のディレクトリから custom_wrapper.conf ファイルを開きます。

   Software AG_directory \profiles\IS_*instance_name*\configuration

   instance_name は Integration Server インスタンスの名前です。

2. 次のプロパティを custom_wrapper.conf に追加します。

   `wrapper.app.parameter.n=switch`

   ここで、*n* は、ファイル内の `wrapper.app.parameter` プロパティの次の未使用の連番で、*switch* はスイッチコマンドです。

   `wrapper.app.parameter.n=switch_parameter`

   ここで、*n* は連番で、*switch_parameter* はスイッチの値です。

   ほとんどのスイッチでは、シーケンス内の次のプロパティとして、custom_wrapper.conf にプロパティを追加する必要があります。

   たとえば、デフォルトのポート番号を 8080 に変更する場合は、custom_wrapper.conf に次を入力します。

   ```
   wrapper.app.parameter.7=-port
   wrapper.app.parameter.8=8080
   ```

   `wrapper.app.parameter` プロパティの詳細については、『*Software AG Infrastructure Administrator's Guide*』を参照してください。

   次の表で、設定ファイル内の設定を変更するのに使用できるスイッチについて説明します。

switch	説明

`-port`	サーバが HTTP 要求を受信待機するポートを指定します。custom_wrapper.conf に `-port` スイッチの `wrapper.app.parameter` プロパティを追加する場合、使用する TCP/IP ポート番号を指定するための `wrapper.app.parameter` プロパティも追加する必要があります。
	
	最初の `wrapper.app.parameter` プロパティで `-port` スイッチを使用することにより、ポートを変更することを指定し、次の `wrapper.app.parameter` プロパティで使用するポート番号を指定します。次に例を示します。
	`wrapper.app.parameter.7=-port`
	`wrapper.app.parameter.8=8080`

switch	説明
	注: ポート番号を変更する場合は、以下の点に留意してください。 ■ このスイッチ変数は、watt.server.port に割り当てられている値を上書きします。 ■ -port スイッチは、新しい HTTP ポートを WmRoot パッケージに永続的に追加します。この新しいポートは、プライマリポートとして追加され、デフォルト値が含まれます。WmRoot パッケージに同じ TCP/IP 番号のポートが既に存在する場合、-port スイッチはその設定を新しいデフォルト値で上書きします。実際には、既存のポートを削除してから新しいポートをデフォルト設定で追加します。 ■ 80 番ポート (HTTP の標準ポート) や 443 番ポート (HTTPS の標準ポート) を使用する場合、UNIX では、「root ユーザ」として実行する必要があります。セキュリティ上の理由により、上位ポート (たとえば、HTTP 用に 5555 番ポート、HTTPS 用に 8080 番ポート) を使用することをお勧めします。必要に応じてファイアウォールを使用し、80 番ポートを適切なポートに再マッピングしてください。ポートの再マッピングの詳細については、27 ページの「アーキテクチャ」を参照してください。
-debug	このセッションのサーバログに記録するデバッグの詳細レベルを指定します。-debug スイッチは、[**ログ**] > [ログ設定] > [サーバロガーの詳細の表示] ページでデフォルト機能として指定され、watt.debug.level に割り当てられた値を上書きします。 custom_wrapper.conf に -debug スイッチの wrapper.app.parameter プロパティを追加する場合、ログに記録する詳細のレベルを指定するための wrapper.app.parameter プロパティも追加する必要があります。以下のレベルを指定できます。

指定内容	記録されるメッセージ
Fatal	重大メッセージのみ
Error	エラーメッセージおよび重大メッセージ
Warn	警告メッセージ、エラーメッセージ、および重大メッセージ
Info	情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ
Debug	デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ

switch	説明	
	Trace	トレースメッセージ、デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ

最初の wrapper.app.parameter プロパティで -debug スイッチを指定することにより、デバッグレベルを変更することを指定し、次の wrapper.app.parameter プロパティでログに記録する詳細のレベルを指定します。次に例を示します。

```
wrapper.app.parameter.11=-debug
wrapper.app.parameter.12=fatal
```

注:
Integration Server 7.1 よりも前の Integration Server では、数値ベースのシステムを使用してサーバログに記録するデバッグ情報のレベルを設定していました。Integration Server では、このシステムとの下位互換性が維持されています。数値ベースのログレベルの詳細については、1095 ページの「サーバ設定パラメータ」の watt.debug.level プロパティの説明を参照してください。

-log

このセッションのサーバログ情報を記録する場所を指定します。custom_wrapper.conf に -log スイッチの wrapper.app.parameter プロパティを追加する場合、ログ情報の保存場所を指定するための wrapper.app.parameter プロパティも追加する必要があります。以下の場所を指定できます。

オプション	説明
filename	このセッションのサーバログ情報を記録するファイルへの完全修飾パスまたは相対パスを指定します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* \instances*instance_name* に相対的なパスです。 *filename* にはディレクトリとファイル名を指定する必要があります。 デフォルトの記録先は watt.debug.logfile サーバ設定パラメータによって制御されます。
none	サーバログ情報をコンピュータの画面 (STDOUT) に書き込みます。
both	サーバログ情報をコンピュータ画面 (STDOUT) および watt.debug.logfile パラ

switch	説明

	メータで指定された記録先に書き込みます。
	
	最初の wrapper.app.parameter プロパティで -log スイッチを指定することにより、ログ情報の保存先を変更することを指定し、次の wrapper.app.parameter プロパティでホームディレクトリの絶対パスまたは none を指定します。次に例を示します。
	`wrapper.app.parameter.13=-log`
	`wrapper.app.parameter.14=none`
	
	注:
	セッション中、*filename* または none のスイッチの値は watt.debug.logfile に割り当てられた値を上書きします。
	
	注:
	none または both の -log スイッチの値は、Microservices Runtime ライセンスを備えている Microservices Runtime や Integration Server がどこに設定変数ログを書き込むかも指定します。ただし、Microservices Runtime はファイル名の値を無視します。ファイル名を指定した場合、Microservices Runtime は設定変数ログファイルを以下の場所だけに書き込みます。 *Integration Server_directory* /instances/*instanceName*/logs/configurationvariables.log

-quiesce	Integration Server を休止モードで起動することを指定します。custom_wrapper.conf に -quiesce スイッチの wrapper.app.parameter プロパティを追加すると、Integration Server は休止モードで起動します。-quiesce スイッチには追加の wrapper.app.parameter プロパティは必要ありません。次に例を示します。 `wrapper.app.parameter.15=-quiesce` 休止モードの詳細については、983 ページの「保守のためのサーバの休止」を参照してください。

3. custom_wrapper.conf 内で、前の手順で追加した wrapper.app.parameter プロパティの合計数を反映させるよう、wrapper.app.parameter.2 プロパティを更新します。

   たとえば、wrapper.app.parameter.2 が 4 (デフォルト) に設定されていて、2 つの wrapper.app.parameter プロパティを追加した場合、wrapper.app.parameter.2 の値を 2 増やします。編集後の wrapper.app.parameter.2 は次のようになります。

   ```
   wrapper.app.parameter.2=6
   ```

   wrapper.app.parameter プロパティの詳細については、『*Software AG Infrastructure Administrator's Guide*』を参照してください。

4. custom_wrapper.conf を保存して閉じます。

Integration Server への Java システムプロパティの受け渡し

custom_wrapper.conf ファイルを変更することで、Integration Server に Java システムプロパティを渡せます。

注:
Microservices Runtime では custom_wrapper.conf を使用しません。Microservices Runtime の場合、 *Integration Server_directory* /bin/setenv.bat(sh) ファイルで JAVA_CUSTOM_OPTS プロパティを使用して Java システムプロパティを設定します。

Integration Server に Java システムプロパティを渡すには

1. custom_wrapper.conf ファイルをテキストエディタで開きます。custom_wrapper.conf ファイルは次のディレクトリにあります。

   Software AG_directory \profiles\IS_*instance_name*\configuration

2. `wrapper.java.additional.n` プロパティを追加します。このプロパティは、Integration Server に渡すプロパティ名と値を指定します。*n* は一意のシーケンス番号です。プロパティ名の先頭に `-D` を付加する必要があります。

   たとえば、custom_wrapper.conf ファイル内の `wrapper.java.additional` プロパティは、次のようになります。

   ```
   wrapper.java.additional.11=-Dmy.prop1=value1
   wrapper.java.additional.12=-Dmy.prop2=value2
   ```

   custom_wrapper.conf ファイル内での値の設定の詳細については、『*Software AG Infrastructure Administrator's Guide*』を参照してください。

3. ファイルを保存して閉じます。

4. サーバを再起動して、変更事項を反映させます。

Integration Server のシャットダウン

サーバーをシャットダウンして Integration Server とすべてのアクティブなセッションを停止できます。この作業は、Integration Server Administrator またはコマンドプロンプトから実行できます。

Microservices Runtime のシャットダウンの詳細については、『 *Developing Microservices with webMethods Microservices Runtime*』を参照してください。

Integration Server Administrator から Integration Server のシャットダウン

Integration Server Administrator から、Integration Server およびすべてのアクティブなセッションをシャットダウンするには、以下の手順に従います。

サーバをシャットダウンするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. > [**シャットダウンまたは再起動**] をクリックします。
3. サーバを直ちにシャットダウンするか、後でシャットダウンするかを選択します。

   [すべてのクライアントセッションが終了した後]Integration Server を何分後にシャットダウンするか指定します。このオプションを選択すると、ユーザアクティビティの監視が開始され、すべての非管理者セッションが終了するか、指定した時間が経過するかいずれか早い方の時間に、サーバが自動的にシャットダウンします。

   [即時]このオプションを選択すると、サーバとすべてのアクティブなセッションは直ちに終了します。
4. アクティブなセッションを表示する方法については、54 ページの「アクティブなセッションの表示」を参照してください。
5. [**シャットダウン**] をクリックします。

Integration Server を Windows からシャットダウンする

Integration Server のインスタンスとすべてのアクティブなセッションを Windows 起動メニューからシャットダウンできます。

注:
ユーザアカウント制御セキュリティ機能を有効にして Windows Server 2008 r2 オペレーティングシステムまたは Windows Server 2012 を実行している場合、Integration Server は完全な Administrator 特権でシャットダウンする必要があります。完全な Administrator 特権で Integration Server をシャットダウンするには、**Software AG** 用のプログラムまたはアプリケーションのリストに移動します。たとえば、Windows Server 2008 の場合、[すべてのプログラム] > [**Software AG**] に移動します。[*instanceName* の停止] を右クリックし、[管理者として実行] オプションを選択します。オペレーティングシステムに Administrator 特権でログインしていない場合、Administrator クレデンシャルの入力が求められます。

Integration Server を Windows からシャットダウンするには

1. [**スタート**] をクリックします。

2. [**すべてのプログラム**] メニューで、 [**Software AG**] フォルダをクリックします。

3. [**サーバの停止**] フォルダをクリックします。

4. [**停止**] をクリックします。**Integration Server**

5. [**Integration Server *instanceName* の停止**]をクリックします。

コマンドプロンプトからの Integration Server のシャットダウン

コマンドプロンプトからサーバとすべてのアクティブなセッションをシャットダウンするには、以下の手順に従います。

コマンドプロンプトからサーバをシャットダウンするには

1. コマンドプロンプトに次のコマンドを入力して、サーバインスタンスのホームディレクトリに切り替えます。

   ```
   cd Software AG_directory\profiles\IS_instance_name\bin
   ```

   instance_name は Integration Server インスタンスの名前です。

   注:
   Integration Server_directory \instances*instance_name*\bin ディレクトリに含まれていた startup.bat/sh と shutdown.bat/sh は廃止されています。Software AG では、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリ内のスクリプトの使用をお勧めします。Command Central から Integration Server を管理する場合は、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリにあるスクリプトを使用する必要があります。

2. 次のコマンドを入力して、サーバを停止します。

Windows の場合:	`shutdown.bat`
UNIX の場合:	`shutdown.sh`

アクティブなセッションの表示

Integration Server が動作している場合は、現在アクティブなセッションを表示できます。

アクティブなセッションを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [統計情報] に移動します。

3. 現在のセッションの数をクリックします。

セッションの強制終了

現在実行中のセッションを除き、単独のセッションまたはすべてのセッションを強制終了できます。

セッションを強制終了するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [統計情報] に移動します。
3. [**すべてのセッション**] フィールドの [**現在**] 列で、現在のセッション数をクリックします。

   Integration Server が [**セッション**] ページを表示します。
4. 強制終了するセッションの [強制終了] 列で、✕をクリックします。

   注:
   または、[**現在のセッション**」の上の [**ユーザセッションを除きすべてのセッションを終了する**] をクリックすると、現在実行しているセッション以外のすべてのセッションを強制終了できます。

   Integration Server は、Integration Server データベースに登録されている、作業中のセッションではない現在のセッションを強制終了します。Integration Server はアクティブな JMS トリガセッションを強制終了しません。[強制終了] 列の ✕ アイコンは、アクティブな JMS トリガセッションに対しては、セッションが期限切れになるまで無効になっています。

Integration Server プロセス ID の表示

Integration Server の各インスタンスは、オペレーティングシステム上で個別の JVM プロセスとして実行します。Integration Server の特定のインスタンスのスレッドダンプの実行や、特定のインスタンスの停止が必要な場合があります (Windows タスクマネージャなどを使用)。これらの操作を行うには、特定の操作を行う対象の Integration Server プロセスのプロセス ID (またはサーバプロセス ID) を確認する必要があります。サーバプロセス ID は、Integration Server Administrator の [製品情報] ページで確認できます。

プロセス ID を表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. > [**製品情報**] をクリックします。

3. [**サーバ環境**] 領域の [**サーバのプロセス ID**] フィールドが表示されるまで下にスクロールします。

Integration Server の再起動

Integration Server を停止して再ロードする必要がある場合は、サーバを再起動します。次のような場合に、サーバを再起動する必要があります。

- **特定の設定を変更した場合**設定を変更した場合、その変更を有効にするために、サーバの再起動が必要になる場合があります。設定の変更を反映するためにサーバの再起動が必要となる変更が行われた場合は、Integration Server Administrator により通知が表示されます。
- **動的に再ロードできない更新サービスを取り込む場合**一般的に、Java 以外のサービスを使用している場合に再起動が必要となります。

サーバを再起動するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. > [**シャットダウンまたは再起動**] をクリックします。
3. サーバを直ちに再起動するか、後で再起動するかを選択します。
   - **[すべてのクライアントセッションが終了した後]**Integration Server を何分後に再起動するか指定します。このオプションを選択すると、ユーザアクティビティの監視が開始され、すべての非管理者セッションが終了するか、指定した時間が経過するかのいずれか早い方の時間に、サーバが自動的に再起動します。
   - **[即時]**このオプションを選択すると、サーバとすべてのアクティブなセッションは直ちに終了します。その後、サーバが再起動します。

     アクティブなセッションを表示する方法については、54 ページの「アクティブなセッションの表示」を参照してください。
4. [**再起動**] をクリックします。

サーバの回復

ハードウェアまたはソフトウェアの問題が原因で Integration Server に障害が発生した場合は、通常の起動手順を使用してサーバを再起動します。サーバはクリーンアップおよび初期化プロセスを実行して、オペレーティング環境のリセットを試行します。

回復プロセスの一環として、サーバは自動的に次のことを実行します。

- キャッシュ環境を再ロードして、障害発生前の状態に戻します。

- Transaction Manager の保証付きデリバーキューを復元します。保証付きデリバーを回復するオプションの詳細については、789 ページの「保証付きデリバーの設定」を参照してください。

サイトで作成したサービスによっては、そのサービスに固有の回復要件がある場合があります。それらの要件については、開発者にお問い合わせください。

状況によっては、サーバを再起動するために、手動操作が必要となる場合があります。

ヒント:
Integration Server を再起動する前に、診断データを収集して実行時の問題をトラブルシュートすることができます。診断ポートとユーティリティの使用方法の詳細については、993 ページの「Integration Server の診断」を参照してください。また、サーバの速度低下または無応答の原因をトラブルシュートするためにスレッドダンプを生成する場合の詳細についても、この章を参照してください。

Integration Server 設定ファイルへの未適用の変更

設定ファイル (*.cnf) に影響する変更を行っているときに Integration Server が不適切にシャットダウンされた場合、Integration Server の再起動時に次のメッセージが表示されることがあります。

```
[ISS.0025.0070C] Integration Server detected files in config/work directory which suggests unapplied configuration changes.
```

Integration Server では、設定ファイルの変更を保存するとき、最初に *Integration Server_directory* /instances/*instanceName*/config/work ディレクトリの一時ファイルに設定の変更を保存します。Integration Server では、一時ファイルに変更内容を保存した後、一時ファイルを実際の設定ファイルに移動します。これは、設定の変更が最初に一時ファイルに保存される前に予期しない動作が発生した場合に、一時ファイルのみが影響を受けるようにするためです。実際の設定ファイルは破損しません。Integration Server の起動時に *Integration Server_directory* /instances/*instanceName*/config/work にファイルが検出された場合、設定ファイルへの変更は一時ファイルに保存されておらず、したがって、実際の設定ファイルは変更されていないことになります。 *Integration Server_directory* /instances/*instanceName*/config/work ディレクトリの内容を使用して、どの設定ファイルが変更途中であったかを確認します。設定ファイルの変更をやり直すかどうかを決定します。やり直す場合、ディレクトリ下のファイルを削除し、設定の変更をやり直します。

Integration Server のデータの完全性および回復能力についての考慮事項

Integration Server では、webMethods 物理記憶領域テクノロジを使用して、重要な操作データが残存 (パーシスト) するようにしています。この記憶領域テクノロジは、データベースタイプの場合と同様に、ログとトランザクション管理のテクノロジを採用しています。通常の操作では、この機能によって、ファイルにパーシストされたデータの整合性、一貫性、および回復能力が維持されています。しかし、このような保護対策にもかかわらず、サーバの異常なシャットダウンや致命的なエラーが発生して、ファイルが回復不可能な状態になる場合があります。

Integration Server Administrator を使用せずに Integration Server をシャットダウンすると、重要なデータファイルが回復不可能な状態になる可能性があります。このような場合に Integration Server を再起動するには、破損したデータファイルを手動で削除または回復する以外に方法はありません。

重要:
重要なデータファイルを保護するために、サーバの設置場所ごとにバックアップおよび復元手続きを確立してください。

Integration Server のデータファイルに保存されているデータは基幹業務に不可欠であるため、障害時の回復に備えて、定期的にバックアップしておく必要があります。あらゆる重要なデータリソースの場合と同様、Integration Server のデータファイルは物理的な障害によって破損する可能性があります。そのような状況でデータを回復するには、破損したデータファイルを最新のバックアップファイルに置き換える以外に方法はありません。バックアップの頻度と種類は、保存するデータの重要度によって異なります。データファイルのバックアップは、Integration Server をアイドル状態にするかまたはシャットダウンして、ディスクアクティビティが存在しない状態でオフラインで実行します。

重要:
Integration Server の重要なデータファイルを定期的にバックアップするために、設置場所ごとに独自の手続きを確立してください。ファイルシステムのバックアップには、任意のバックアップユーティリティを使用できます。バックアッププロセスを実行するときには、Integration Server をシャットダウンするかまたは休止状態にして、ディスクアクティビティが存在しない状態にする必要があります。この制約によって、バックアップ時に重要なデータファイルが一貫した状態で確実にキャプチャされます。Integration Server がアクティブな状態でバックアップを実行すると、キャプチャされるデータファイルのスナップショットの一貫性が失われてしまい、バックアップの回復目的に使用できなくなる可能性があります。

Integration Server の重要なデータファイル

Integration Server の現在の作業ディレクトリには、重要なデータファイルが含まれている 2 つのサブディレクトリがあり、そのサブディレクトリ内のファイルは、回復時に使用するためにバックアップしておく必要があります。重要な 3 つのサブディレクトリについて、以下に説明します。

- ./DocumentStore

  このサブディレクトリ内のファイルには、Integration Server によって処理される、ローカルにパーシストされるドキュメントが含まれています。これらのファイルが失われると、パーシストされているドキュメントを損失する可能性があります。次の 6 つのファイルをバックアップしておいてください。

  ISResubmitStoredata0000000

  ISResubmitStorelog0000000

  ISTransStoredata0000000

  ISTransStorelog0000000

  TriggerStoredata0000000

TriggerStorelog0000000

- ./WmRepository4

  このサブディレクトリ内のファイルには、Integration Server のメタデータが含まれています。これらのファイルが失われると、設定情報を損失し、手動による再設定が必要になる可能性があります。次の 2 つのファイルをバックアップしておいてください。

  FSDdata0000000

  FSDlog0000000

Java サービスラッパー

webMethods Integration Server は Software AG Common Platform で実行され、これは Java 仮想マシン (JVM) で実行されます。Java Service Wrapper は、Tanuki Software, Ltd. によって開発されたアプリケーションです。これは、Integration Server を実行する JVM を起動するユーティリティプログラムです。

JVM の起動に加えて、Java Service Wrapperには JVM の監視機能、コンソール出力のログ機能、およびスレッドダンプの生成機能が備わっています。ここでは、Integration Server で Java Service Wrapperの機能を使用する方法について説明します。Java Service Wrapperの概要については、webMethods 製品共通のマニュアル『*Software AG Infrastructure Administrator's Guide*』を参照してください。

注:
Microservices Runtime では Java Service Wrapper を使用しません。

Java Service Wrapper設定ファイル

Integration Server の場合、Java Service Wrapperの設定ファイルは、次のディレクトリにあります。

Software AG_directory \profiles\IS_*instanceName*\configuration

Integration Server の起動時に、次のファイルのプロパティ設定により、JVM の設定と Java Service Wrapperのログ機能および監視機能の動作が決定されます。

ファイル名	説明
wrapper.conf	Integration Server によってインストールされたプロパティ設定を含みます。 **重要:** Software AG に依頼された場合を除いて、このファイルの内容を変更しないでください。
custom_wrapper.conf	wrapper.conf 内のインストール済み設定を変更するプロパティを含みます。

ファイル名	説明
	Java Service Wrapperのプロパティ設定を変更する必要がある場合、このファイル内で変更を行います。 **注:** Integration Server バージョン 9.7 以降、Integration Server は setenv.bat/sh または server.bat/sh から設定を取得しません。Integration Server は起動時に、custom_wrapper.conf からすべてのクラスパスの変更を取得します。Integration Server がクラスパスを構成する方法の詳細については、966 ページの「Integration Server クラスパスの設定方法」を参照してください。

ここでは、Integration Server が Java Service Wrapperのためにサポートする設定の変更について説明します。Integration Server の場合、Java Service Wrapper に対してここで説明する設定変更以外の変更は加えないでください。

JVM 設定

Java Service Wrapper が JVM を起動すると、メモリ設定やクラスパス内のディレクトリなどを指定する設定が提供されます。Integration Server を起動すると、Java Service Wrapper は、*Software AG_directory* \profiles\IS_*instance_name*\configuration フォルダにある wrapper.conf ファイルおよび custom_wrapper.conf ファイルからこれらの設定を取得します。

Integration Server のデフォルトのプロパティ設定を変更する必要がある場合は、*Software AG_directory* \profiles\IS_*instance_name*\configuration にある custom_wrapper.conf ファイルを使用して、設定を変更できます。Integration Server の custom_wrapper.conf プロパティの設定の詳細については、123 ページの「サーバの設定」を参照してください。Java Service Wrapper の一般情報については、『*Software AG Infrastructure Administrator's Guide*』を参照してください。

ラッパーログ

Java Service Wrapperは、コンソール出力をログファイルに記録します。ログには、ラッパー自身によって、および Integration Server を実行している JVM によってコンソールに送信された出力が含まれます。ラッパーログは、特に Windows サービスとして Integration Server を実行するときに役立ちます。これは、通常、コンソール出力がこのモードでは使用できないからです。

Integration Server の Java Service Wrapperログは、次のファイルに記録されます。

Software AG_directory \profiles\IS_*instanceName*\logs\wrapper.log

ログを表示するには、テキストエディタでログファイルを開きます。

ログ記録のプロパティ

ラッパー設定ファイル内の wrapper.console および wrapper.log プロパティにより、ラッパーログの内容、形式、および動作が決定されます。

Integration Server がインストールするログ設定は、多くの環境に適しています。ただし、インストール済みの設定が環境の要件を満たしていない場合、以下のプロパティを変更できます。手順とその他の情報については、webMethods 製品共通のマニュアル『*Software AG Infrastructure Administrator's Guide*』を参照してください。

プロパティ	値
wrapper.console.format	コンソールに表示されるメッセージの形式。
wrapper.console.loglevel	コンソールに表示される詳細のレベル。
wrapper.logfile	Integration Server のコンソール出力が記録されるファイル。
wrapper.logfile.format	ラッパーログファイルに記録されるメッセージの形式。
wrapper.logfile.loglevel	ラッパーログファイルに記録される詳細のレベル。この設定は、wrapper.console.loglevel 内の設定以下のレベルにする必要があります。
wrapper.logfile.maxsize	ログの最大サイズ。
wrapper.logfile.maxfiles	Java Service Wrapperが保持する古いログの数。
wrapper.syslog.loglevel	Windows システムのイベントログまたは UNIX システムの syslog に記録されるメッセージ。

障害の監視

Java Service Wrapperは、特定の条件について JVM を監視し、その条件を検出すると JVM を再起動するかその他のアクションを実行します。

次の表では、Integration Server で使用したり、ユーザが設定したりできる障害監視機能について説明しています。これらの機能の詳細については、webMethods 製品共通のマニュアル『*Software AG Infrastructure Administrator's Guide*』を参照してください。

機能	有効であるか	ユーザ設定の可否
JVM タイムアウト	はい	いいえ。wrapper.ping プロパティは、Software AG から指示がない限り変更しないでください。
デッドロック検出	いいえ	いいえ。この機能は有効にしないでください。
コンソールフィルタリング	いいえ	いいえ。この機能は有効にしないでください。

スレッドダンプの生成

Java Service Wrapperは、JVM のスレッドダンプを生成するユーティリティを提供します。スレッドダンプは、スレッドのブロックまたはデッドロックを引き起こす可能性のある、スレッドの競合の問題を見つけるのに役立ちます。

Java ラッパーサービスを使用してスレッドダンプを生成する方法については、webMethods 製品共通のマニュアル『*Software AG Infrastructure Administrator's Guide*』を参照してください。

4 複数の Integration Server インスタンスの実行

- Integration Server インスタンスの概要 64
- 同じマシン上で複数の Integration Server インスタンスを実行する際のガイドライン .. 64
- 新規 Integration Server インスタンスの作成について 64
- is_instance スクリプトについて 65
- 新規 Integration Server インスタンスの作成 66
- Integration Server インスタンスの更新 70
- サーバインスタンスでのパッケージの更新 71
- サーバインスタンスのデータベースプロパティの更新 72
- サーバインスタンスからパッケージの削除 73
- サーバインスタンスでの Language Pack の更新 73
- サーバインスタンスの削除 75

Integration Server インスタンスの概要

webMethods Integration Server が物理的に 1 つインストールされているホストマシンで、Integration Server の複数インスタンスを作成および実行することができます。これは、*webMethods Integration Server Clustering Guide*で説明するクラスタリングの場合と異なります。同じホストマシンで Integration Server の複数インスタンスを実行するときは、サーバごとにパッケージ、設定ファイル、ログファイル、および更新があります。しかし、クラスタリングとは異なり、すべてのサーバインスタンスが共通ライブラリを共有できます。共通ライブラリを共有すると、修正やパッチを各サーバにインストールするときのオーバーヘッドや作業の量が減ります。

各サーバインスタンスは、Integration Server Administrator を使用して個別に管理します。また、特定の管理、設定、および監視タスクを 1 か所から Software AG Command Central を使用して実行することもできます。Software AG Command Central を使用して Integration Server インスタンスを管理する方法については、Command Central のマニュアルを参照してください。

注:
Microservices Runtime は複数のインスタンスの実行をサポートしていません。

同じマシン上で複数の Integration Server インスタンスを実行する際のガイドライン

同じマシン上で複数の Integration Server インスタンスを実行する場合は、以下のガイドラインが適用されます。

- 共通ライブラリは SoftwareAG\common および *Integration Server_directory* \lib にあります。これらのライブラリ内のファイルは、すべてのサーバインスタンスで使用できます。たとえば、カスタム jar ファイルを *Integration Server_directory* \lib\jars\custom ディレクトリ内に配置することで、これらのファイルをすべてのサーバインスタンスが利用できるようにすることができます。
- インスタンスレベルで存在する更新およびカスタム jar は、共通ライブラリ内に存在する更新およびカスタム jar よりも優先します。
- 各サーバインスタンスの各ポート番号は、ホストマシンに 1 つ以上の NIC がある場合を除き、一意であることが必要です。ポートには、プライマリポート、診断ポート、および167 ページの「ポートの設定」で説明するその他すべてのポートが含まれます。
- 複数の Integration Server インスタンスの作成や実行には、追加のライセンスは必要ありません。ただし、追加のサーバインスタンスの作成や実行により、同時実行プロセスの数が増加します。追加のサーバインスタンスを作成する前に、システムリソースおよび Software AG ライセンスをチェックしてください。

新規 Integration Server インスタンスの作成について

Integration Server インスタンス作成スクリプト is_instance.bat/sh (*Integration Server_directory* \instances ディレクトリにある) を使用して、新しいIntegration Server インスタンスを作成することができます。

新しい Integration Server インスタンスを作成するときは、OSGI コンポーネントが格納されている SoftwareAG\profiles\IS_*instance_name* ディレクトリ内に、スクリプトによって新しいフォルダが作成されます。

また、インスタンスのホームディレクトリも、 *Integration Server_directory* \instances*instance_name* に作成されます。ホームディレクトリには、696 ページの「事前定義済みのパッケージ」で説明する事前定義済みパッケージが含まれます。

サーバインスタンスで他のパッケージに含まれるコンポーネントまたはサービスを使用するには、パッケージを Integration Server に追加する必要があります。たとえば、サーバインスタンスで webMethods Deployer、webMethods API Gateway、または webMethods Application Platform を使用するには、WmDeployer、WmAPIGateway、または WmAppPlat パッケージをそれぞれ Integration Server インスタンスに追加する必要があります。Software AG Installer はパッケージリポジトリにパッケージをインストールします。パッケージリポジトリは *Software AG_directory* \IntegrationServer\packages ディレクトリにあります。

新しい Integration Server インスタンスを作成する方法については、66 ページの「新規 Integration Server インスタンスの作成」を参照してください。パッケージをサーバインスタンスに追加する方法については、71 ページの「サーバインスタンスでのパッケージの更新」を参照してください。

is_instance スクリプトについて

is_instance.bat/sh スクリプトは、Integration Server インスタンスを作成、更新、および削除します。また、Integration Server インスタンスでパッケージおよび Language Pack をインストール、更新、または削除し、Integration Server インスタンスによって使用されるデータベースのプロパティを更新します。

is_instance.bat/sh スクリプトは、以下のディレクトリにあります。

Integration Server_directory `\instances`

シンタックス

コマンドプロンプトで、以下のシンタックスを使用します。

Integration Server_directory `\instances\is_instance.bat` *command parameters*

コマンドシンタックスが正しくない場合、スクリプトはエラー情報をコンソールに書き込みます。

is_instance スクリプトコマンド

is_instance.bat/sh スクリプトで使用できるコマンドの説明を以下の表に示します。

コマンド	説明
`create`	Integration Server の新規インスタンスを作成します。このパラメータの使用方法の詳細については、66 ページの「新規

コマンド	説明
	Integration Server インスタンスの作成」を参照してください。
`update`	指定されたパッケージのリストを Integration Server インスタンスに追加し、Integration Server インスタンスによって使用されるデータベースのプロパティを更新します。このパラメータの使用方法の詳細については、71 ページの「サーバインスタンスでのパッケージの更新」および72 ページの「サーバインスタンスのデータベースプロパティの更新」を参照してください。
`deletePackages`	パッケージを Integration Server インスタンスから削除します。このパラメータの使用方法の詳細については、73 ページの「サーバインスタンスからパッケージの削除」を参照してください。
`updateLanguagePack`	Integration Server インスタンスの Language Pack ファイルを更新します。73 ページの「サーバインスタンスからパッケージの削除」.
`delete`	Integration Server インスタンスを削除します。このパラメータの使用方法の詳細については、75 ページの「サーバインスタンスの削除」を参照してください。

新規 Integration Server インスタンスの作成

新しい Integration Server インスタンスを作成するには、`create` コマンドを使用します。

新しいインスタンスを作成する前に、64 ページの「同じマシン上で複数の Integration Server インスタンスを実行する際のガイドライン」および 64 ページの「新規 Integration Server インスタンスの作成について」で情報を確認してください。

新しい Integration Server インスタンスを作成するには

1. 次のディレクトリに移動します。

   Integration Server_directory `\instances`

2. 以下のようにして、インスタンススクリプトを実行します。

```
is_instance.bat/sh create
-Dinstance.name=instance_name
-Dadmin.password=Administrator_password
-Dadmin.changeOnLogin=change_password
-Dprimary.port=primary_port_number
-Dinstance.ip=default_bind_address
-Ddiagnostic.port=diagnostic_port_number
-Dsecure.port=secure_port_number
-Djmx.port=jmx_port_number
```

```
-Dlicense.file=license_file_location
-Dinstall.service=install_service
-Ddb.alias=database_alias_name
-Ddb.type=database_type
-Ddb.username=database_username
-Ddb.password=database_password
-Ddb.url=database_URL
-Dpackage.list=package_list
```

変数	指定内容
instance_name	新しいインスタンスの一意の名前。
Administrator_password	オプション。Administrator ユーザに使用するパスワード。Administrator ユーザのパスワードをここで設定しないと、Administrator ユーザのパスワードは Integration Server のインストール時に指定されたデフォルト値になります。
change_password	オプション。新しいインスタンスへのログインが成功したときに、Administrator ユーザのパスワードを変更するかどうか。パスワードの変更が必要な場合、`yes` に設定します。それ以外の場合は、`no` に設定します。`admin.changeOnLogin` の値を指定しない場合、新しいインスタンスでは Integration Server のインストール時に設定されたログイン時のパスワード変更設定がデフォルトになります。つまり、Integration Server のインストール担当者が、最初のログイン成功時に Administrator ユーザのパスワードの変更を要求するというオプションを選択した場合、新しい Integration Server インスタンスでも同様の操作が要求されます。
primary_port_number	オプション。新しいインスタンスのデフォルト HTTP ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は `5555` です。
default_bind_address	オプション。Integration Server のこのインスタンスでポートをバインドする先のデフォルト IP アドレス。マシンに複数の IP アドレスがあり、そのマシンで Integration Server の異なるインスタンス間で同じポート番号を使用する場合、バインドアドレスを指定します。この場合、各インスタンスには異なるバインドアドレスを指定する必要があります。デフォルトのバインドアドレスを指定しない場合、ポートはすべてのネットワークインタフェースを受信待機し、複数のインタスタンスで同じポート番号が使用されません。 **注:** Integration Server では、指定された *default_bind_address* 値を watt.server.inetaddress 設定プロパティの値として使用します。

変数	指定内容
diagnostic_port_number	オプション。新しいインスタンスのデフォルト診断ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は `9999` です。
secure_port_number	オプション。新しいインスタンスのデフォルト HTTPS ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は `5543` です。
jmx_port_number	オプション。新しいインスタンスのデフォルト JMX ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は `8075` です。
license_file_location	オプション。Integration Server ライセンスファイルの場所。 スクリプトによって、ライセンスファイルが指定された場所から IntegrationServer/instances/*instance_name*/config/licenseKey.xml ファイルにコピーされます。 このパラメータはオプションですが、ライセンスファイルの場所が指定されない場合、Integration Server インスタンスは 30 分後にシャットダウンします。 **注:** 場所名にスペースが含まれている場合、場所名を引用符 (”) で囲む必要があります。次に例を示します。 `-Dlicense.file=”license file location”`
install_service	オプション。インスタンスにサービスを登録するかどうか。デフォルトは「`false`」です。 **注:** このオプションは、Windows プラットフォームのみで有効です。
database_alias_name	オプション。データベースのエイリアス名。組み込みデータベースのデフォルト値は `Embedded Database Pool` であり、外部データベースの場合は `JDBC_POOL_ALIAS` です。
database_type	オプション。データベースのタイプ。次のいずれかのデータベースを指定します。 ■ `sql`: Microsoft SQL Server ■ `oracle`: Oracle ■ `db2`: DB2

変数	指定内容
	これらの値は、大文字と小文字を区別しません。このパラメータが指定されていない場合、新しい Integration Server インスタンスでは、デフォルトである組み込みの IS 内部データベースを使用します。
database_username	オプション。データベースに接続するユーザ名。
database_password	オプション。データベースユーザのパスワード。
database_URL	オプション。データベースの接続 URL。 次に例を示します。 `jdbc:wm:oracle://<server>:<1521|port>;serviceName=<value>[;<option>=<value>...]`
package_list	オプション。サーバインスタンスに追加するパッケージのカンマ区切りリスト。たとえば、「`WmPRT,WmTN`」と入力します。 指定するパッケージは webMethods Integration Server パッケージリポジトリの一部である必要があります。Software AG Installer はパッケージリポジトリにパッケージをインストールします。パッケージリポジトリは *Software AG_directory* \IntegrationServer\packages ディレクトリにあります。 `all` を指定すると、 *Integration Server_directory* \packages ディレクトリの webMethods Integration Server パッケージリポジトリ内にある、事前定義済み以外のすべてのパッケージを追加します。事前定義済みのパッケージのリストについては、696 ページの「事前定義済みのパッケージ」を参照してください。 **注:** サーバインスタンスで他のパッケージに含まれるコンポーネントまたはサービスを使用するには、パッケージを Integration Server インスタンスに追加する必要があります。たとえば、サーバインスタンスで webMethods Deployer、webMethods API Gateway、または webMethods Application Platform を使用するには、WmDeployer、WmAPIGateway、または WmAppPlat パッケージをそれぞれ Integration Server インスタンスに追加する必要があります。

3. オプションで、サーバインスタンスの作成後にサーバインスタンスにパッケージを追加します。手順については、71 ページの「サーバインスタンスでのパッケージの更新」を参照してください。

4. *Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリにある startup.bat/sh ファイルを実行して、Integration Server インスタンスを開始します。

注:
Integration Server_directory \instances*instance_name*\bin ディレクトリに含まれていた startup.bat/sh と shutdown.bat/sh は廃止されています。Software AG では、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリ内のスクリプトの使用をお勧めします。Command Central から Integration Server を管理する場合は、*Software AG_directory* \profiles\IS_*instance_name*\bin ディレクトリにあるスクリプトを使用する必要があります。

注:
webMethods Application Platform 用のパッケージ WmAppPlat を追加した場合、新しい Integration Server インスタンスには、Tomcat の設定されたインスタンスが含まれます。この Tomcat インスタンスはデフォルトの HTTP および HTTPS ポートとしてそれぞれポート 8072 と 8074 を使用します。これらのデフォルトポートは、Integration Server のデフォルトインスタンスで Tomcat によって使用されるポートと競合します。Command Central を使用して、Integration Server の新しいインスタンスで Tomcat 用のデフォルトの HTTP および HTTPS ポート番号を変更する必要があります。webMethods Application Platform での Integration Server インスタンス作成の詳細については、*webMethods Application Platform User's Guide*を参照してください。

Integration Server インスタンスの更新

既存のパッケージを更新したり、サーバインスタンスでその他のパッケージをインストールしたりするには、`update` コマンドを使用します。また、Integration Server がオフラインモードである場合にデータベースプロパティを更新するときにも使用します。

Integration Server インスタンスを更新するには

1. 以下のようにして、is_instance スクリプトを実行します。

```
is_instance.bat update -Dinstance.name=instance_name
-Ddb.alias=database_alias_name
-Ddb.type=database_type
-Ddb.username=database_username
-Ddb.password=database_password
-Ddb.url=database_URL
-Dpackage.list=package_list
```

詳細については、71 ページの「サーバインスタンスでのパッケージの更新」および72 ページの「サーバインスタンスのデータベースプロパティの更新」を参照してください。

注:
インストーラを使用してパッケージをインストールまたは更新する場合、インストーラによって作成された最初のインスタンスにパッケージをインストールするオプションがあります。このオプションを選択すると、パッケージはインストール中にインスタンスに自動でコピーされます。そのため、インスタンスで `update` コマンドを実行する必要はありません。

サーバインスタンスでのパッケージの更新

既存のパッケージを更新したり、サーバインスタンスでその他のパッケージをインストールしたりするには、`update` コマンドを使用します。

Integration Server インスタンスでパッケージを更新するには

1. Integration Server インスタンスを停止します。
2. 次のディレクトリに移動します。

   Integration Server_directory `\instances`

3. 以下のようにして、is_instance スクリプトを実行します。

```
is_instance.bat update -Dinstance.name=instance_name   -Dpackage.list=package_list
```

変数	指定内容
instance_name	パッケージを追加するインスタンスの名前。
package_list	サーバインスタンスに追加するパッケージのカンマ区切りリスト。たとえば、「`WmPRT,WmTN`」と入力します。 指定するパッケージは webMethods Integration Server パッケージリポジトリの一部である必要があります。Software AG Installer はパッケージリポジトリにパッケージをインストールします。パッケージリポジトリは *Software AG_directory* \IntegrationServer\packages ディレクトリにあります。 `all` を指定すると、 *Integration Server_directory* \packages ディレクトリの webMethods Integration Server パッケージリポジトリ内にある、事前定義済み以外のすべてのパッケージを追加します。事前定義済みのパッケージのリストについては、696 ページの「事前定義済みのパッケージ」を参照してください。 **注:** サーバインスタンスで他のパッケージに含まれるコンポーネントまたはサービスを使用するには、パッケージをインスタンスに追加する必要があります。たとえば、サーバインスタンスで webMethods Deployer、webMethods API Gateway、または webMethods Application Platform を使用するには、WmDeployer、WmAPIGateway、または WmAppPlat パッケージをそれぞれインスタンスに追加する必要があります。

スクリプトによって、パッケージが *Integration Server_directory* \instances*instance_name*\packages ディレクトリに追加されます。

サーバインスタンスのデータベースプロパティの更新

Integration Server がオフラインモードである場合にデータベースプロパティを更新するには、`update` コマンドを使用します。

注:
Integration Server インスタンスで埋め込みデータベースを使用している場合、データベースプロパティを更新することはできません。

Integration Server インスタンスのデータベースプロパティを更新するには

1. Integration Server インスタンスを停止します。
2. 次のディレクトリに移動します。

   Integration Server_directory `\instances`

3. 以下のようにして、is_instance スクリプトを実行します。

```
is_instance.bat update -Dinstance.name=instance_name
-Ddb.alias=database_alias_name
-Ddb.type=database_type
-Ddb.username=database_username
-Ddb.password=database_password
-Ddb.url=database_URL
```

変数	指定内容
instance_name	データベースプロパティを更新するインスタンスの名前。
database_alias_name	データベースのエイリアス名。
database_type	オプション。データベースのタイプ。 次のいずれかのデータベースを指定します。 ■ *sql*: Microsoft SQL Server ■ *oracle*: Oracle ■ *db2*: DB2 これらの値は、大文字と小文字を区別しません。
database_username	オプション。データベースに接続するユーザ名。このパラメータは、*database_type* を指定する場合にのみ必須です。
database_password	オプション。データベースユーザのパスワード。このパラメータは、*database_type* を指定する場合にのみ必須です。

変数	指定内容
database_URL	オプション。データベースの接続 URL。このパラメータは、*database_type* を指定する場合にのみ必須です。

注:
再起動時に、データベース設定が正しくないために Integration Server インスタンスがデータベースに接続できない場合、Integration Server は前のデータベース設定に戻ります。

サーバインスタンスからパッケージの削除

`deletePackages` コマンドは、指定されたパッケージまたはパッケージのリストをサーバインスタンスから削除します。

Integration Server インスタンスからパッケージを削除するには

1. Integration Server インスタンスを停止します。
2. 次のディレクトリに移動します。

   Integration Server_directory `\instances`

3. 以下のようにして、is_instance スクリプトを実行します。

   ```
   is_instance.bat deletePackages -Dinstance.name=instance_name -Dpackage.list=package_list
   ```

変数	指定内容
instance_name	パッケージを削除するインスタンスの名前。
package_list	Integration Server インスタンスから削除するパッケージのカンマ区切りリスト。たとえば、「`packageA,packageB`」とします。

is_instance スクリプトによって、 *Integration Server_directory* \instances*instance_name*\packages*package_name* ディレクトリにある指定されたパッケージが削除されます。

サーバインスタンスでの Language Pack の更新

webMethods Integration Server で Language Pack をインストール、変更、または削除したら、各サーバインスタンスで `updateLanguagePack` コマンドを実行して、ファイルを同期および反映する必要があります。

注:
インストーラを使用して Language Pack をインストールまたは更新する場合、インストーラによって作成されたインスタンスにパッケージをインストールするオプションがあります。このオプションを選択すると、Language Pack はインストール中にインスタンスに自動で同

期されます。そのため、インストーラによって作成されたインスタンスで `updateLanguagePack` コマンドを実行する必要はありません。

Integration Server インスタンスで Language Pack ファイルを更新するには

1. Integration Server インスタンスを停止します。
2. 次のディレクトリに移動します。

   Integration Server_directory `\instances`

3. 以下のようにして、is_instance スクリプトを実行します。

```
is_instance.bat updateLanguagePack -Dinstance.name=instance_name
                -Dinstall.mode=mode -Dpackage.list=pacakge_list
                -Dfeature.list=feature_list
```

変数	指定内容
instance_name	更新するインスタンスの名前。
mode	スクリプトによってインスタンスで Language Pack ファイル (jar および DSP、HTML、クラスなど) を追加するのか削除するのかを示す処理モード。次のいずれかを指定します。 ■ `install`。Language Pack ファイルを追加または更新する場合。このオプションが指定された場合、Language Pack ファイルがスクリプトによってパッケージリポジトリから指定されたインスタンスにコピーされます。 **注:** インスタンスを更新する前に、Software AG Installer を使用して Language Pack をインストールしてください。ファイルがパッケージリポジトリに存在しない場合、スクリプトによってインスタンスを更新できません。 ■ `uninstall`。Language Pack ファイルを削除する場合。このオプションが指定された場合、Language Pack ファイルがスクリプトによってインスタンスから削除されます。 **注:** `-Dinstall.mode=uninstall` を実行する前に、Software AG Installer を使用して Language Pack をアンインストールしてください。そうしないと、ファイルがインスタンスから削除されません。
package_list	オプション。Language Pack の追加先パッケージのカンマ区切りリスト。

変数	指定内容
	注: `-Dpackage.list` を指定しない場合、`-Dfeature.list` を指定する必要があります。ただし、`-Dpackage.list` と `-Dfeature.list` は両方指定できます。`-Dinstall.mode` が `install` に設定されている場合にのみ、`-Dpackage.list` と `-Dfeature.list` のどちらか一方または両方を指定する必要があります。
	たとえば、API Gateway として機能する Integration Server インスタンスの Language Pack をインストールするには、次のように指定します。 `-Dpackage.list=WmAPIGateway`
feature_list	オプション。Language Pack の追加先機能のカンマ区切りリスト。機能のリストについては、製品マニュアルを確認してください。
	注: `-Dfeature.list` を指定しない場合、`-Dpackage.list` を指定する必要があります。ただし、`-Dpackage.list` と `-Dfeature.list` は両方指定できます。`-Dinstall.mode` が `install` に設定されている場合にのみ、`-Dpackage.list` と `-Dfeature.list` のどちらか一方または両方を指定する必要があります。

サーバインスタンスの削除

`delete` コマンドは、 *Integration Server_directory* \instances*instance_name* および SoftwareAG\profiles\IS_*instance_name* ディレクトリにあるすべてのファイルおよびフォルダを削除します。

重要:
サーバインスタンスを削除すると、そのファイルやディレクトリ (カスタムディレクトリを含む) を取得することができなくなります。

Integration Server インスタンスを削除するには

1. Integration Server インスタンスを停止します。
2. 次のディレクトリに移動します。

   Integration Server_directory `\instances`
3. 以下のようにして、is_instance スクリプトを実行します。

```
is_instance.bat delete -Dinstance.name=instance_name
```

変数	指定内容
instance_name	削除するインスタンスの名前。

5 Integration Server Administrator の使用

- Integration Server Administrator とは 78
- Integration Server Administrator の起動 78
- Windows での Integration Server Administrator の起動 79
- My webMethods Server から Integration Server Administrator へのアクセス 79
- webMethods Integration Server Administrator の概要 80
- レガシー Integration Server Administrator へのアクセス 82
- 新しい Integration Server Administrator における名前とナビゲーションの変更 83
- サーバ設定ファイル 84
- Integration Server Administrator からのログオフ 84

Integration Server Administrator とは

Integration Server Administrator は、webMethods Integration Server の管理に使用するブラウザベースのアプリケーションです。このユーティリティを使用して、サーバアクティビティの監視、ユーザアカウントの管理、パフォーマンスの調整および操作パラメータの設定を行うことができます。

ネットワークに接続され、ブラウザがインストールされているワークステーションであれば、どのワークステーションからでも Integration Server Administrator を実行できます(Integration Server Administrator は、サービスを使用して作業を遂行するブラウザベースのアプリケーションです)。

Integration Server Administrator の起動

Integration Server Administrator を使用するには、ブラウザを開いて、Integration Server インスタンスが稼動しているホストおよびポートを指定します。

Integration Server Administrator を使用するには、サーバが稼動している必要があります。サーバが稼動していないと、ブラウザに次のようなエラーメッセージが示されます。

```
「このページは表示できません。」
「サーバへの接続を確立できませんでした。」
```

Integration Server Administrator を 起動するには

1. ブラウザを開きます。

2. ブラウザのアドレスバーに、Integration Server インスタンスが稼動しているホストおよびポートを入力します。

   例

   Integration Server Administrator を実行しているマシン上のデフォルトのポートでサーバが稼動している場合は、次のように入力します。

   ```
   http://localhost:5555/
   ```

   サーバが QUICKSILVER と呼ばれるマシン上のポート 4040 で 稼動している場合は、次のように入力します。

   ```
   http://QUICKSILVER:4040/
   ```

3. Administrator 特権を持つユーザ名とパスワードでサーバにログオンします。

   初めて Administrator ユーザが Integration Server Administrator にログインする場合は、以下の点に留意してください。

   - Integration Server のインストール時に、インストールの担当者が Administrator ユーザのパスワードを指定します。このパスワードを Integration Server への最初のログイン時に指定する必要があります。

- インストールの担当者は、Integration Server Administrator へのログインが成功した直後に Administrator のパスワードの変更を要求するオプションを設定することもあります。設定した場合、Administratorが最初にログインした後、Integration Server Administrator に [新規パスワードを作成する] ダイアログボックスが表示されます。

重要:
ユーザ名とパスワードは大文字と小文字が区別されるので、文字の組み合わせをそのまま正確に入力します。

Windows での Integration Server Administrator の起動

Windows で Integration Server Administrator を起動するには

1. [**スタート**] をクリックします。
2. [**スタート**] メニューで、Integration Server がインストールされたフォルダを開きます。
3. [**Integration Server *instanceName releaseName*** の開始] をクリックします。

   たとえば、[Integration Server default 10.11 の開始] です。
4. Integration Server が稼働したら、[**スタート**] メニューで [**Intergration Server Administrator for *instanceName releaseName***] をクリックします。
5. Administrator 特権を持つユーザ名とパスワードでサーバにログオンします。

   初めて Administrator ユーザが Integration Server Administrator にログインする場合は、以下の点に留意してください。

   - Integration Server のインストール時に、インストールの担当者が Administrator ユーザのパスワードを指定します。このパスワードは Integration Server への最初のログイン時に指定する必要があります。
   - インストールの担当者は、Integration Server Administrator へのログインが成功するとすぐに Administrator のパスワードの変更を要求するオプションを設定することもあります。設定した場合、Administratorが最初にログインした後、Integration Server Administrator に [新規パスワードを作成する] ダイアログボックスが表示されます。

重要:
ユーザ名とパスワードは大文字と小文字が区別されるので、文字の組み合わせをそのまま正確に入力します。

My webMethods Server から Integration Server Administrator へのアクセス

My webMethods Server で作業している場合は、My webMethods の ESB 管理ウィンドウからも Integration Server Administrator にアクセスできます。Integration Server Administrator を ESB 管理ウィンドウに追加するには、以下の手順に従います。

Integration Server Administrator を My webMethods Server から使用できるようにするには

1. My webMethods Server から、[**ナビゲーション**] > [アプリケーション] > [管理] > [統合] > [ESB] に移動します。

2. [**サーバの追加...**] をクリックして、各フィールドに次のように入力します。

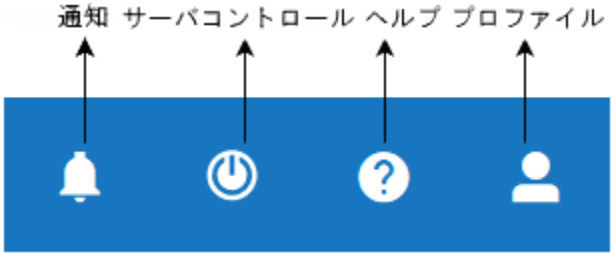
フィールド	入力内容
説明	Integration Server の簡単な説明。
[サーバホスト]	My webMethods から管理する Integration Server のホストマシン。
[サーバポート]	My webMethods Server が要求を送信する先の Integration Server ポート。
[シングルサインオン]	My webMethods Server のシングルサインオン機能を使用すると、My webMethods Server にログオンするユーザは Integration Server にログオンしなくてもこれを管理できるようになります。シングルサインオン機能の設定手順については、556 ページの「My webMethods のシングルサインオン認証 」を参照してください。
[セキュア接続の使用]	Integration Server への接続に My webMethods Server で SSL を使用するかどうか。 **注:** このオプションを選択できるのは、SSL および認証を使用するように Integration Server が設定されている場合のみです。詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
[色]	インタフェースの周囲に使用する色を選択します。複数の Integration Server がある場合に、この機能を使用すると、それぞれの Integration Server を互いに区別できます。たとえば、実稼動環境の Integration Server にはオレンジ色を、テスト環境の Integration Server には青色を使用することができます。

3. [**OK**] をクリックします。

webMethods Integration Server Administrator の概要

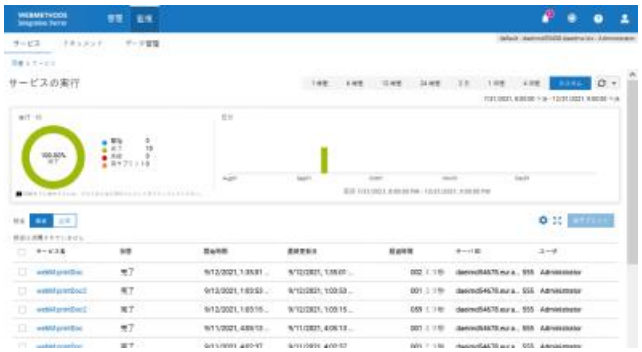
Integration Server Administrator では、再設計されたタブベースのインタフェースを通じてすべての管理タスクにアクセスできます。また、新しいインタフェースは、バージョン 10.11 以降、デフォルトです。Integration Server Administrator を起動すると、ブラウザに [**ダッシュボード**] ページが表示されます。ページの左側のナビゲーションパネルには、Integration Server の管理に必要な関連するコンポーネントと設定をグループ化するメニューが表示されます。

上部ナビゲーションバーから以下の項目にアクセスできます。



- [**通知**]: Integration Server 内のイベントに関する情報を提供し、ビジネスに不可欠なイベントへの対処に役立つシステム通知。たとえば、認証の期限切れ、サーバの再起動が必要な設定の変更などのイベントの通知を受け取ります。現在、通知の読み取り、個々の通知またはすべての通知の無視、および通知のサブスクライブまたはアンサブスクライブを行うことができます。
- [**サーバコントロール**]: 休止モードの開始または終了、サーバのシャットダウンまたは再起動などのタスク。
- [**ヘルプ**]: オンラインヘルプおよび製品の [**製品情報**] ページ。Integration Server Administrator のページの [**ヘルプ**] アイコンをクリックすることで、そのページに関する情報を取得できます。ヘルプシステムは、すべてのページの設定とデータに関する説明を提供します。
- [**プロファイル**]: 現在のユーザのロケール設定および [**ログアウト**] オプション。ロケール設定を変更したり、デフォルトに戻したりできます。現在のセッションを続行する必要がなくなった場合は、Integration Server Administrator からログアウトできます。ログアウトすると、セッションがクリーンアップされ、ブラウザ内のキャッシュが消去されます。

[**監視**]: [監視] タブでは、 フローサービスおよびコーディングされたサービス (Java サービスなど) を監視できます。 webMethods Broker クライアントによってパブリッシュまたはサブスクライブされたドキュメント、インダウトドキュメント、失敗したドキュメント、またはトリガーの再試行の上限に達したドキュメントを監視できます。また、サービスおよびドキュメントのエラーの詳細を 取得することもできます。さらに、サービスパイプラインとドキュメントを編集し、 再サブミットすることもできます。



サービス監視の最大パフォーマンスを確認するため、過去の期間の監視済みデータをアーカイブしたり削除することができます。詳細については、『*Service Monitoring in Integration Server Guide*』を参照してください。

注:
休止設定で監視を無効にした後、サーバを休止した場合、監視は一時停止されます。

[**共通ディレクトリサービス**]: [管理] タブの左側ナビゲーションから CDS (Common Directory Services: 共通ディレクトリサービス) にアクセスできます。CDS は、異なるユーザディレクトリに接続し、ユーザ、グループ、役割、および webMethods リソースへのユーザのアクセスを管理する統一されたメカニズムを提供します。詳細については、PDF ドキュメント『*Managing Users with Common Directory Services*』を参照してください。

注:
Integration Server Administrator で共通ディレクトリサービスへのアクセスを有効にするには、Integration Server インストールの一部として [**Common Directory Server Support**] および [**Central User Management**] コンポーネントを選択します。また、インストール時またはその後に、データベース接続を設定する必要もあります。詳細については、『*Installing Software AG Products On Premises*』ガイドを参照してください。

レガシー Integration Server Administrator へのアクセス

レガシー Integration Server Administrator に一時的に切り替えたり、デフォルトインタフェースとして使用したりすることができます。

次の手順では、レガシー Integration Server Administrator に一時的に切り替えることができます。新規またはデフォルトの Integration Server Administrator に切り替える場合、[**新しい管理**

コンソールを試す] をクリックします。ただし、Integration Server を起動するごとにレガシー Integration Server Administrator を使用する場合、`watt.server.defaultPackage` パラメータの値を `WmRoot` に設定し、サーバを再起動します。デフォルトで、このパラメータは `WmAdmin` に設定されます。この場合、新しいデフォルトの Integration Server Administrator が開きます。

レガシー Integration Server Administrator に切り替えるには

1. Integration Server Administrator にログオンします。
2. [**パッケージ**] に移動します。
3. [パッケージの一覧] で、WmRoot パッケージの隣にある をクリックします。新しいタブにレガシー Integration Server Administrator が開きます。

新しい Integration Server Administrator における名前とナビゲーションの変更

このマニュアルでは、新しい Integration Server Administrator であるデフォルトインタフェースについて説明しています。したがって、このマニュアルでは、レガシーインタフェースに適用できるナビゲーション手順を説明していません。ただし、以前のバージョンの Integration Server Administrator に精通しているユーザに対し、このトピックでは新しい Integration Server Administrator における名前とナビゲーションの変更をまとめています。

バージョン 10.5 以前	バージョン 10.7 以降
[休止モードを開始する] または [休止モードを終了する]	> [休止モードを開始する] または [休止モードを終了する]
[シャットダウンと再起動]	> [シャットダウンまたは再起動]
[ヘルプ]	各ページの アイコン
[製品情報]	> [製品情報]
[ログオフ]	> [ログアウト]
[サーバ] > [スケジューラ]	[サーバ] > [スケジューリング]
[セキュリティ] > [ポート]	[サーバ] > [ポート]
[設定] > [ライセンス]	[サーバ] > [ライセンス]
[設定] > [ログ]	[ログ] > [ログ設定]
[設定] > [メッセージング]	[メッセージング]

バージョン 10.5 以前	バージョン 10.7 以降
[設定] > [プロキシサーバ]	[外部サーバ] > [プロキシサーバ]
[設定] > [休止]	[サーバ] > [休止]
[設定] > [リモートサーバ]	[外部サーバ] > [リモートサーバ]
[設定] > [SFTP]	[外部サーバ] > [SFTP]
[設定] > [メタデータ]	[Asset Publisher] > [設定] このページを表示するには、[**パッケージ**] > [管理] > [WmAssetPublisher] に移動し、次に アイコンをクリックします。

サーバ設定ファイル

Integration Server の設定は `server.cnf` サーバ設定ファイル内に保存されています。このファイルは `Integration Server_directory\instances\instance_name\config` ディレクトリに保存され、サーバの動作を規定するパラメータが記載されます。

通常は、Integration Server Administrator を使用して `server.cnf` ファイルのパラメータを設定します。ただし、テキストエディタを使用してファイルを直接編集する必要がある場合もあります。

`server.cnf` ファイル内のパラメータの一覧とそのデフォルト値については、1095 ページの「サーバ設定パラメータ」を参照してください。

Integration Server Administrator からのログオフ

現在のセッションを続行する必要がなくなった場合は、Integration Server Administrator からログアウトします。ログアウトすると、セッションがクリーンアップされ、ブラウザ内のキャッシュが消去されます。

Integration Server Administrator からログアウトするには

- > [**ログアウト**] をクリックします。

6 ダッシュボードの使用

- 概要 86
- [JVM メトリクス] 88
- [使用状況] 90
- サービス 92
- [API] 97
- ダッシュボードの設定 98

Integration Server Administrator には、主要なシステムおよびサーバのメトリクスを読みやすい形式で表示するダッシュボードが含まれています。総合的なダッシュボードは、Integration Server のシステムステータス、ヘルス情報、使用パターン、および全体的なパフォーマンスを監視するのに役立ちます。ダッシュボードには、[**サーバ**] > [**統計情報**] ページで利用できるメトリクスやその他の多くのメトリクスが含まれます。

ダッシュボードでは、さまざまなカテゴリのメトリクス用に以下のタブを使用します。

- [概要]
- [JVM メトリクス]
- [使用状況]
- [サービス]
- [API]

ダッシュボードのデータは選択した期間に対するものです。ダッシュボードの上部に表示される事前定義済みの期間の 1 つから選択するか、[**カスタム**] を使用して独自に期間を設定することができます。

ダッシュボードには、チャートと表を使用してメトリクスが表示されます。表をカスタマイズして、必要な列を表示できます。

概要

[概要] タブには、現在の Integration Server インスタンスの主要なシステムプロパティが表示されます。

[概要] タブには、情報ボックスとチャートを使用して、以下の情報が表示されます。

- [**バージョン**]: Integration Server インスタンスの完全なバージョンを表示します。
- [**ハードウェア**]: プロセッサのタイプおよびマシンで利用可能なメモリの総容量を表示します。
- [**経過時間**]: 現在の Integration Server インスタンスが起動した以降の時間を時と分で表示します。
- [**ディスクスペース**]: このチャートは以下の情報を表示します。
  - [合計]: 現在の Integration Server インスタンスがインストールされているディスクの総容量。
  - [使用中]: 現在の Integration Server インスタンスがインストールされているディスクの使用容量。

  [**ディスクスペース**] カードの赤いセクションには、ディスクスペース使用の [**しきい値**] が表示されます。`watt.server.dashboard.diskspace.threshold` サーバ設定パラメータが、しきい値を決定します。詳細については、1136 ページの「watt.server.」の `watt.server.dashboard.diskspace.threshold` を参照してください。
- [**メモリ**]: このチャートは以下の情報を表示します。
  - [使用中]: Java 仮想マシン (JVM) で使用されているメモリ容量を示します。
  - [合計]: Java 仮想マシン (JVM) が取得できるメモリの最大容量を示します。
  - [コミット済み]: Java 仮想マシンによる使用が保証されているメモリ容量を示します。

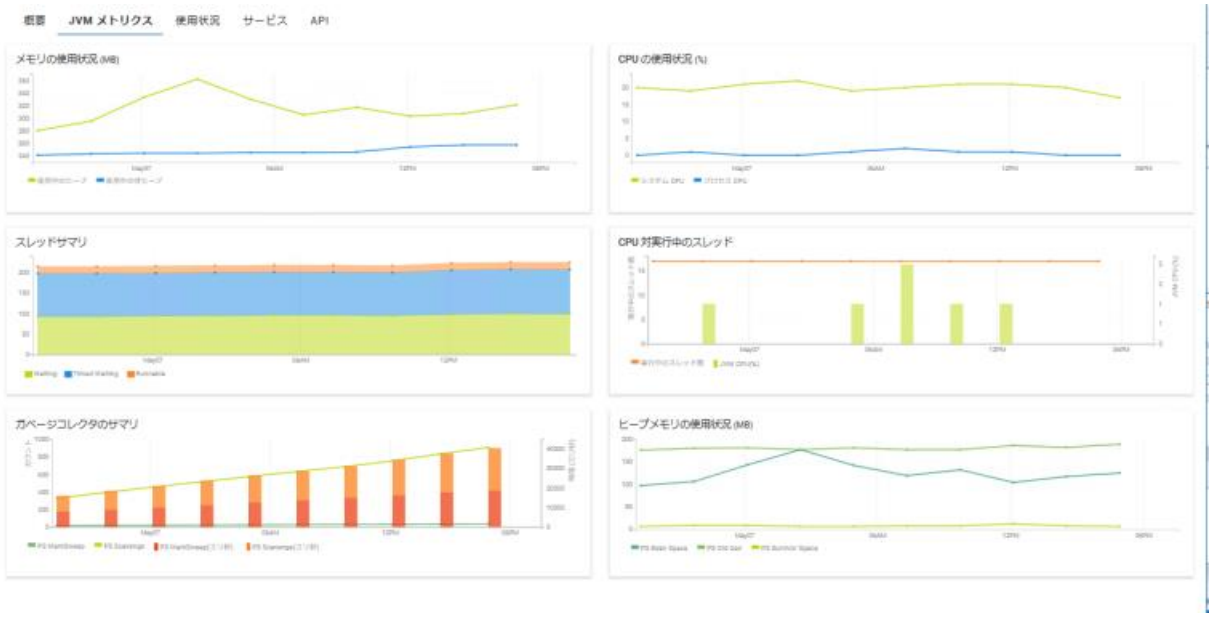
  [**メモリ**] カードの赤いセクションには、メモリ使用の [**しきい値**] が表示されます。`watt.server.dashboard.memory.threshold` サーバ設定パラメータが、しきい値を決定します。詳細については、1136 ページの「watt.server.」の `watt.server.dashboard.memory.threshold` を参照してください。
- [**CPU**]: Java 仮想マシンの最近の CPU 使用状況。
- [**クラスタの状態**]:
  - クラスタのステータス: 現在の Integration Server インスタンスがクラスタの一部である場合、クラスタのステータスは [オン] です。それ以外の場合、[オフ] です。
  - クラスタの状態: クラスタの状態に従って、以下のメッセージの 1 つが表示されます。

- クラスタは動いています。
- クラスタが動いていません。
- クラスタ化されたインスタンスではありません

[JVM メトリクス]

[JVM メトリクス] タブでは、Integration Server が実行されている JVM のヒープメモリ、スレッドサマリおよびガーベッジコレクションの詳細を表示できます。また、時系列での CPU とメモリの使用状況など、オペレーティングシステムのリソース情報を表示することもできます。

注:
[JVM メトリクス] タブには、Integration Server の現在のインスタンスからのデータのみ表示されます。このデータは `java.lang.management.MemoryPoolMXBean` クラスによってレポートされるデータです。



JVM メトリクスは以下のグラフを使用して表示されます。

- **[メモリの使用状況]**
  - ヒープメモリ (使用中): ヒープで使用されているメモリの容量 (MB 単位) を示します。
  - 非ヒープメモリ: 非ヒープメモリは、JVM がクラスのフィールドとメソッド、メソッドコード、ランタイム定数プール、インターンされた文字列などのクラスレベルの情報を保存する場所です。JVM によって使用される非ヒープメモリの容量 (MB 単位) を示します。
- **[CPU の使用状況]**
  - [システム CPU]: システム全体の CPU の使用状況。このメトリクスはパーセンテージ値として表示されます。0% の値は、すべての CPU がアイドル状態であることを示します。100% の値は、監視間隔内にすべての CPU が実行されていたことを示します。

- [プロセス CPU]: JVM プロセスの CPU の使用状況。このメトリクスはパーセンテージ値として表示されます。0% の値は、すべての CPU がアイドル状態であることを示します。100% の値は、監視間隔内にすべての CPU が実行されていたことを示します。

- **[スレッドサマリ]**

  `java.lang.Thread.State` クラスで定義された状態のスレッド数を示します。

  - [Waiting]: 別のスレッドが特定のアクションを実行するまで無期限に待機するスレッドの数。
  - [Timed Waiting]: 別のスレッドが特定のアクションを実行するまで、指定された待機期間、待機するスレッドの数。
  - [Runnable]: JVM で実行されているスレッドの数。

- **[CPU 対実行中のスレッド]**

  - [実行中のスレッド数]: Java 仮想マシンで実行されている (実行可能ステータスの) スレッドの合計数。
  - [JVM CPU (%)]: JVM プロセスの CPU の使用状況。

- **[ガーベッジコレクタのサマリ]**

  注:
  ガーベッジコレクタ (GC) は JVM 固有であり、以下で説明するものとは異なることがあります。PS MarkSweep および PS Scavenge は、以下で説明するように、ほとんどのプラットフォームにおけるデフォルトのガーベッジコレクタ実装 (最も一般的な 2 つ) です。

  - [PS MarkSweep]: 「第 2 世代」メモリで PS MarkSweep ガーベッジコレクタによって完了されたガーベッジコレクションの合計数。
  - [PS Scavenge]: 「第 1 世代」メモリの Eden コンポーネントと Survivor コンポーネントで PS Scavenge ガーベッジコレクタによって完了されたガーベッジコレクションの合計数。
  - [PS MarkSweep (ミリ秒)]: PS MarkSweep ガーベッジコレクタが「第 2 世代」メモリからメモリを解放するためにかかる時間 (ミリ秒単位)。
  - [PS Scavenge (ミリ秒)]: PS Scavenge ガーベッジコレクタが「第 1 世代」メモリの Eden コンポーネントと Survivor コンポーネントからメモリを解放するためにかかる時間 (ミリ秒単位)。

- **[ヒープメモリの使用状況]**

  注:
  ヒープメモリのメモリプールは JVM 固有であり、以下で説明するものとは異なることがあります。一般に使用されるメモリプールについて以下で説明します。

  - [PS Eden 領域]: 「第 1 世代」メモリの Eden コンポーネントで使用されるメモリの容量 (MB 単位)。

- [PS Survivor 領域]: 「第 1 世代」メモリの Survivor コンポーネントで使用されるメモリの容量 (MB 単位)。
- [PS 第 2 世代]: 「第 2 世代」メモリで使用されるメモリの容量 (MB 単位)。

[使用状況]

[使用状況] タブには、サーバの使用状況の統計情報が表示されます。クラスタのすべてのノードからセッションと要求の統合的なデータを表示できます。

- **[セッション]**

  Integration Server のローカルインスタンスがクラスタの一部である場合、[セッション] のグラフにはクラスタのすべてのノードのデータが表示されます。

  注:
  セッションデータには、ユーザセッションおよび Integration Server によって内部的に開始されたセッションが含まれます。

  - [ステートフル]: ポーリング間隔内に開始されたステートフルセッションの数。
  - [ステートレス]: ポーリング間隔内に開始されたステートレスセッションの数。
  - [ライセンス]: ポーリング間隔内に開始されたライセンスセッションの数。
  - [最大]: ポーリング間隔内に開始されたセッションの合計数。

- **[要求数]**

  Integration Server のローカルインスタンスがクラスタの一部である場合、[要求数] のグラフにはクラスタのすべてのノードのデータが表示されます。

  [要求数] メトリクスはトップレベルサービスの呼び出しにのみ関連します。トップレベルサービスによる他のサービスへの要求は含まれません。

  - [成功]: 間隔内に正常に完了した要求数。
  - [エラー]: 間隔内に失敗した要求数。

- [合計]: 間隔内に完了した要求の合計数。要求の合計数は成功とエラーの要求の合計に等しくなります。

- **[セッションサマリ]**

  [経過時間] フィールドには、サーバが起動してから経過した時間が表示されます。テーブルには、以下の統計情報が表示されます。

  - [すべてのセッション]
  - [ライセンスされているセッション]
  - [ステートフルセッション]
  - [サービスインスタンス]
  - [サービススレッド]
  - [システムスレッド]

  テーブルの各行には、以下の内容が表示されます。

  - [現在] 列には、現在アクティブなセッションの数が表示されます。
  - [ピーク] 列には、サーバでそれまでに同時実行されたセッションの最大数が表示されます。
  - [制限] 列には、サーバで同時に実行できるセッションの最大数が表示されます。これは、[ライセンスされているセッション] と [ステートフルセッション] でのみ利用できます。

  注:
  [セッションサマリ] テーブルには、[セッション] グラフとは異なり、ローカルインスタンスからのデータのみ表示されます。[セッション] グラフには、ローカルインスタンスがクラスタ化されている場合、クラスのすべてのノードのデータが表示されます。

- **[要求サマリ]**

  [要求サマリ] メトリクスはトップレベルサービスの呼び出しにのみ関連します。トップレベルサービスから他のサービスへの要求に関連するメトリクスは要求サマリには含まれません。

  さらに、Wm* パッケージのサービスはデフォルトで [要求サマリ] メトリクスから除外されます。`watt.server.stats.packages.exclude` サーバ設定プロパティにパッケージ名を追加することで、[要求サマリ] メトリクスから他のパッケージのサービスを除外できます。`watt.server.stats.packages.exclude` パラメータの詳細については、1136 ページの「watt.server.」を参照してください。

  Integration Server のローカルインスタンスがクラスタの一部である場合、[要求数] のグラフにはクラスタのすべてのノードのデータが表示されます。

  [要求サマリ] テーブルでは、2 つのテーブルに要求メトリクスが表示されます。最初のテーブルには、[現在] 列と [稼動以来] 列があります。[現在] 列には、現在のポーリング間隔のメトリクスが表示されます。[稼動以来] 列には、サーバの起動以来のメトリクスが表示されます。

- [要求の開始]: この行には、対応する期間に開始された要求の数が表示されます。
- [完了済み要求数]: この行には、対応する期間にサーバによって完了された要求の数が表示されます。
- [平均時間 (ミリ秒)]: この行には、対応する期間に要求の完了にかかった平均処理時間 (ミリ秒) が表示されます。
- [サービスエラー]: この行には、エラーログの循環以降、現在のログ間隔に終了を失敗した要求の合計数が表示されます。

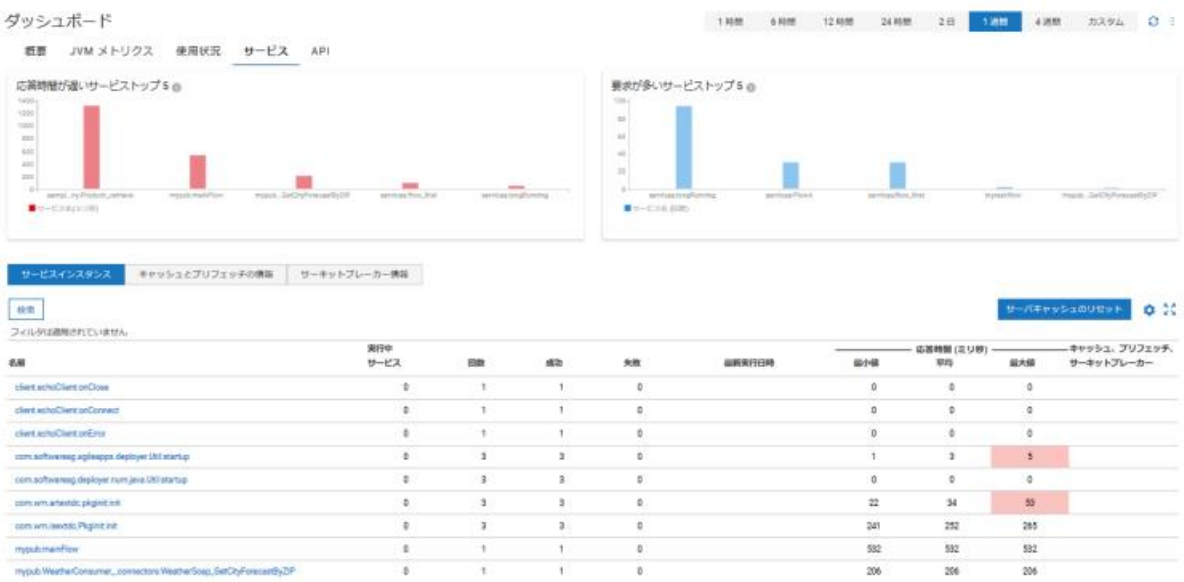
[要求サマリ] の 2 番目のテーブルには、[開始] 列と [完了] 列があります。

- [最後のポーリング間隔中の要求]: この行には、最終ポーリング間隔で開始および完了した要求の数がそれぞれ表示されます。

注:
[要求サマリ] テーブルには、[要求] グラフとは異なり、ローカルインスタンスからのデータのみ表示されます。[要求] グラフには、ローカルインスタンスがクラスタ化されている場合、クラスのすべてのノードのデータが表示されます。

サービス

[サービス] タブには、同じクラスタのすべてのノードからのデータがまとめられています。サービスグラフには、応答時間と要求数によるトップサービスが降順に表示されます。[使用状況] タブの要求数メトリクスと異なり、[サービス] タブのメトリクスにはすべてのレベルのサービスが含まれます。



[サービス] タブには以下のグラフが表示されます。

- [**応答時間が遅いサービストップ 5**]: 一定の時間範囲内で最も平均応答時間が長いサービストップ 5 (システムサービスを除く)。

- **[要求が多いサービストップ 5]**: 一定の時間範囲内で最も要求数が多いサービストップ 5 (システムサービスを除く)。

また、[サービス] タブには、サービスインスタンス、キャッシュとプリフェッチ情報、およびサーキットブレーカー情報に関連する以下のデータも表示されます。

オプション	説明
[検索]	このタブの [検索] コマンドを使用して [サービスインスタンス] タブのテーブルをフィルタできます。これによって、システムサービスおよび現在実行されていないサービスをビューから除外できます。
[サーバキャッシュのリセット]	サーバキャッシュをリセットしてカウントを消去し、すべてのサービスのデータ収集タイマを再起動できます。 特定のサービスのキャッシュをリセットするには、pub.cache.serviceResults:resetServiceCache サービスを使用できます。
[設定]	[設定] メニューを使用して、以下の操作を行うことができます。 ■ 交互の行に異なる背景色を表示する場合、[縞模様] を有効にします。 ■ [表示する列] メニューオプションを使用して、テーブルに表示する列を選択します。

[サービスインスタンス]

[サービスインスタンス] タブには、すべての実行済みサービスおよび現在実行中のサービスに関する情報が表示されます。データは同じクラスタ内のすべてのインスタンスのものです。

注:
`watt.server.stats.packages.exclude` プロパティにリストされたパッケージに属するサービスは [サービスインスタンス] タブに表示されます (タブの [検索] コントロールを使用して除外されている場合を除く)。しかし、サービスの [最新実行日時] 列および [キャッシュ、プリフェッチ、サーキットブレーカー] 列は空になります。

列	説明
[パッケージ]	サービスが入ったパッケージ。 **注:** デフォルトで、この列は表示されません。この列を表示するには、⚙> **[表示する列]** をクリックします。
[名前]	実行中または実行済みのサービスの完全修飾名。
[実行中のサービス]	現在実行中のサービスのインスタンス数。
[回数]	指定された時間内に実行されたこのサービスのインスタンス数。
[成功]	サービスの成功した実行数。

列	説明
[失敗]	失敗に終わったこのサービスの実行数。
[最新実行日時]	サービスが最後に実行された日時を表示します。
[応答時間 (ミリ秒)]	

注:
応答時間は、60 秒など、特定のサンプリング間隔内のすべての応答の平均時間です。

[最小値]:	選択された期間内でサービスの実行にかかった最小時間 (ミリ秒単位)。
[平均]:	選択された期間内でサービスの実行にかかった平均時間 (ミリ秒単位)。
[最大値]:	選択された期間内でサービスの実行にかかった最大時間 (ミリ秒単位)。

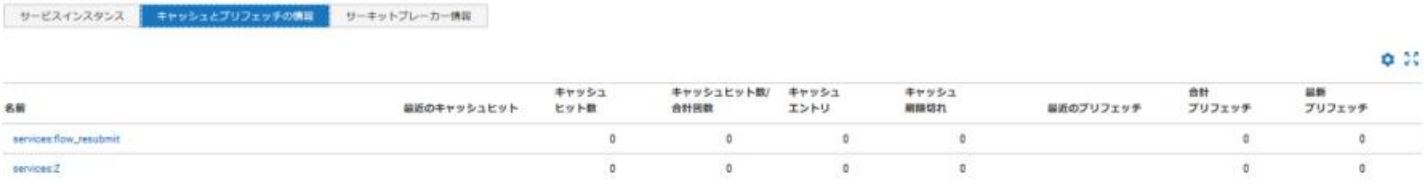
[キャッシュ、プリフェッチ、サーキットブレーカー]	サービスのキャッシュオプションまたはプリフェッチオプションが有効になっているかどうか、またはサービスがサーキットブレーカーであるかどうかを示します。状態に応じて、1 つ以上のアイコンが表示されます。 ■ キャッシュが有効になっていることを示します。 ■ プリフェッチが有効になっていることを示します。 ■ サービスがサーキットブレーカーであることを示します。

`watt.server.dashboard.service.responseTime.threshold` サーバ設定パラメータは、サービスの特定の実行が [サービスインスタンス] タブの最大応答時間に対してフラグを立てるほど遅いかどうかを決定します。詳細については、1136 ページの「watt.server.」の `watt.server.dashboard.service.responseTime.threshold` を参照してください。

[キャッシュとプリフェッチの情報]

このタブには、キャッシュを現在使用しているサービスのキャッシュ使用に関する情報が表示されます。

注:
`watt.server.stats.packages.exclude` プロパティにリストされたパッケージに属するサービスは [キャッシュとプリフェッチの情報] タブに表示されません。



サービスのキャッシュ使用

列	説明
[パッケージ]	サービスが入ったパッケージ。 **注:** デフォルトで、この列は表示されません。この列を表示するには、⚙> **[表示する列]** をクリックします。
[名前]	実行中または実行済みのサービスの完全修飾名。
[最近のキャッシュヒット]	このサービスのキャッシュ済み結果が最後にアクセスされた日時。
[キャッシュヒット数]	このサービスのキャッシュ済み結果がアクセスされた合計回数。
[キャッシュヒット数/合計回数]	キャッシュから結果を抽出することで実行されたサービス要求の割合。
[キャッシュエントリ]	サービスのアクティブなキャッシュエントリ数。
[キャッシュ期限切れ数]	サービスのキャッシュが期限切れになる時刻。
[最近のプリフェッチ]	サーバがサービスの結果を最後にプリフェッチしてキャッシュした日時。 **注:** キャッシュは **[キャッシュ期限切れ]** で指定された正確な時刻にリフレッシュされるとは限りません。キャッシュスイーパスレッドによって、0 ～ 15 秒ずれることがあります。`watt.server.cache.flushMins` プロパティはキャッシュのリフレッシュ時間を制御します。
[合計のプリフェッチ]	サーバがサービスの結果をプリフェッチしてキャッシュした合計回数。
[最新プリフェッチ]	最後のポーリング間隔でサーバがサービスの結果をプリフェッチしてキャッシュした合計回数(ほとんどのサーバで、ポーリング間隔は 10 分に設定されています)。

[サーキットブレーカー情報]

Integration Server では、サーキットブレーカーが設定された各サービスのサーキットブレーカー統計情報を収集します。

注:
`watt.server.stats.packages.exclude` プロパティにリストされたパッケージに属するサービスは [サーキットブレーカー情報] タブに表示されません。

注:
サーキットブレーカー機能は Microservices Runtime にあるサービスでデフォルトで利用できます。Integration Server でサーキットブレーカー機能を使用するには、Integration Server に追加ライセンスが必要です。

サーキットブレーカーがサービスに設定されていない場合、[**サーキットブレーカー情報**] テーブルには「サーキットブレーカーが有効になっているサービスはありません」というメッセージが表示されます。

サーキットブレーカーが設定された各サービスの統計情報

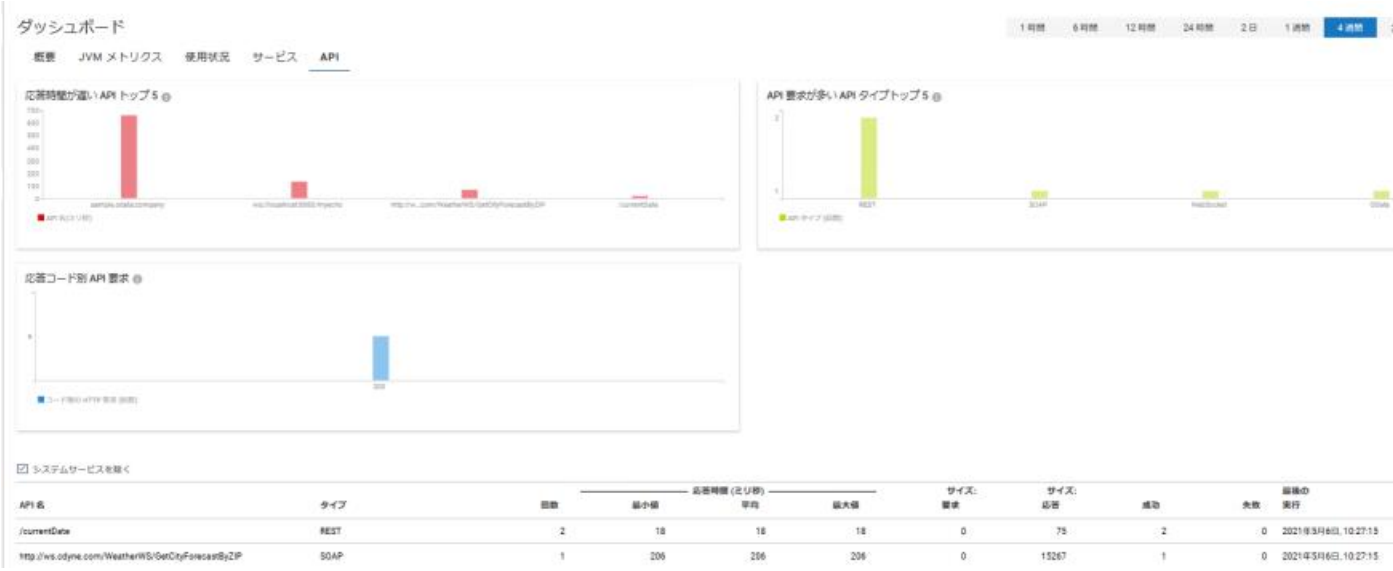
列	説明
[パッケージ]	サービスが入ったパッケージ。 **注:** デフォルトで、この列は表示されません。この列を表示するには、⚙> [**表示する列**] をクリックします。
[名前]	サーキットブレーカーが設定されたサービスの完全修飾名。
[状態]	サービスの状態。 サーキットの状態。サーキットの状態は以下のいずれかになります。 ■ クローズ:サービスは実行されます。 ■ オープン:サービスは実行されません。代わりに、サーキットブレーカー設定に従って、サービスは例外を返すか、代替サービスを実行します。 ■ ハーフオープン:サーキットの状態がハーフオープンに変更になってからのサービスの最初の要求はサービス実行になります。他のすべての要求は待機します。サービスが正常に実行された場合、サーキットは閉じられ、待機している要求が実行されます。サービスが失敗の例外で終了した場合、サーキットは再び開きます。 ■ オープン時間:サーキットが開いた時間。
[オープン時間]	サーキットを閉じる (つまり、実行呼び出しを再試行する) アクションが実行された時間。
[ハーフオープン時間]	実際のサービスの再試行が行われています。成功の場合、サーキットは閉じます。成功しなかった場合、サーキットは開いたままです。
[クローズ時間]	サーキットが最後に閉じられた時間。この時間は成功した最後の要求の実行を表します。

列	説明
[オープン回数]	Integration Server の起動以降にサーキットが開かれた回数。
[要求数]	サーキットが最後に開かれた以降の受信要求数。

[API]

[API] タブには REST、SOAP、ODATA、GRAPHQL、WEBSOCKET の各 API 呼び出しを含む、すべての API 呼び出しに関するデータがまとめられます。

注:
[API] タブには、クラスタのすべてのノードからのデータが表示されます。



[API] タブでは、以下のチャートを使用して、API に関連するメトリックを分類します。

- **[応答時間が遅い API トップ 5]**: 一定の時間範囲内で最も平均応答時間が長い API トップ 5(システムサービスを除く)。
- **[API 要求が多い API タイプトップ 5]**: 一定の時間範囲内で最も要求数が多い API タイプトップ 5(システムサービスを除く)。
- **[応答コード別 API 要求]**: 一定の時間範囲内の HTTP 応答コードの要求数(システムサービスを除く)。

さらに、以下の API 情報がテーブルとして表示されます。

列	説明
[API 名]	API の名前。
[タイプ]	API のタイプ (API_REST、API_ODATA、API_GRAPHQL、API_WEBSOCKET、API_SOAP)。

列	説明
[回数]	選択された時間内の API 呼び出しの合計数。
[応答時間 (ミリ秒)]	
注: 応答時間は、60 秒など、特定のサンプリング間隔内のすべての応答の平均時間です。	
[最小値]:	選択された期間内で API の実行にかかった最小時間 (ミリ秒単位)。
[平均]:	選択された期間内で API の実行にかかった平均時間 (ミリ秒単位)。
[最大値]:	選択された期間内で API の実行にかかった最大時間 (ミリ秒単位)。
[サイズ: 要求]	選択された期間内の API 要求の平均サイズ (バイト単位)。
[サイズ: 応答]	選択された期間内の API 応答の平均サイズ (バイト単位)。
[成功]	成功した API 呼び出しの合計数。
[失敗]	失敗に終わった API 呼び出し。
[最後の実行]	最後の API 実行の日時。

注:
デフォルトで、システムサービスはこのテーブルから除外されます。

ダッシュボードの設定

ダッシュボードに表示される統計情報を収集し保存すると、メモリ、JDBC プール接続、スレッド、ディスク容量などの Integration Server リソースが使用されます。Integration Server Administrator は、ダッシュボードに使用されるリソースを制限するオプションを備えています。

注:
ダッシュボード設定のすべてのオプションは、monitor.cnf 内にある設定パラメータによって設定できます。このファイルは *Integration Server_directory* /instances/*instanceName*/config/monitor/monitor.cnf に置かれています。Statistics Data Collector の設定パラメータの詳細リストについては、を参照してください。

ダッシュボードを設定するには

1. Integration Server Administrator で > [**ダッシュボード設定**] をクリックします。

2. [ダッシュボード設定] ダイアログで、次の設定オプションを指定します。

オプション	説明
[統計データの収集]	Statistics Data Collector が有効かどうか。無効になっている場合、Statistics Data Collector はリソースを使用せず、Integration Server へのオーバーヘッドは加わりません。Integration Server が Enterprise Gateway Server として機能する場合など、一部の環境では、Statistics Data Collector を無効にする必要があります。 Statistics Data Collector が無効になると、ダッシュボードの多くのテーブルやグラフは空になります。 このオプションは monitor.cnf の `enabled` パラメータに対応します。
[結果の最大数]	Statistics Data Collector に許容される結果の最大数。最小値は 1 です。 デフォルト: 1500000 このオプションは monitor.cnf の `maxResults` パラメータに対応します。
[メモリのサイズ (MB)]	Statistics Data Collector 専用にする JVM のメモリサイズ (MB 単位)。最小値は 256 です。 デフォルト: 256 このオプションは monitor.cnf の `luceneMemorySize` パラメータに対応します。
[インデクサースレッドカウント]	同時にインデックス作成できるリソースの数。使用可能なスレッドよりインデックスタスクの方が多い場合、スレッドが使用可能になるまで、タスクは待機します。インデクサースレッドは別のスレッドプールのものです。つまり、インデクサースレッドはサーバスレッドプールのものではありません。最小値は 2 です。 デフォルト: 8 このオプションは monitor.cnf の `indexerThreadCount` パラメータに対応します。
[最大 JDBC プール接続]	Statistics Data Collector によって使用可能な ISInternal 機能で使用される JDBC プールからの同時 JDBC 接続数。最小値は 4 です。 デフォルト: 4 **注:**

オプション	説明
	ISInternal 機能によって使用される JDBC プールエイリアスには、Statistics Data Collector で使用できる接続数をサポートするだけの [**接続数の最大値**] が必要です。 このオプションは monitor.cnf の `maxJdbcPoolConnections` パラメータに対応します。
[サーチャースレッドカウント]	同時に実行できる検索要求数。検索要求数がこの値を超える場合、現在実行中の検索要求が完了するまで、ペンディングの検索要求はブロックされます。最小値は 2 です。 デフォルト: 8 このオプションは monitor.cnf の `searcherThreadCount` パラメータに対応します。
[保持する日数]	Statistics Data Collector によって使用されるデータベースのデータを保持する最大日数。最小値は 1 です。 デフォルト: 30 このオプションは monitor.cnf の `retentionDays` パラメータに対応します。

3. [**保存**] をクリックします。Integration Server はダッシュボード設定を再ロードします。

注:
ダッシュボード設定を変更しても、Statistics Data Collector は再起動されません。ただし、Statistics Data Collector を無効にした場合、Statistics Data Collector はシャットダウンし、すべてのアクティビティを停止します。

7 ユーザとグループの管理

- ユーザとグループ 102
- ユーザアカウントの定義 102
- ユーザアカウントの無効化と有効化 115
- グループの定義 116

ユーザとグループ

ユーザとグループの情報をサーバに定義するには、Integration Server Administrator を使用します。ユーザの定義には、ユーザ名、パスワードおよびグループメンバーシップの情報が含まれます。グループの定義には、グループ名とグループ内のユーザ一覧が含まれます。ユーザとグループの情報は、サーバが保存および管理します。

または、サイトでユーザとグループの情報として次に示す外部ディレクトリのいずれかを使用している場合は、外部ディレクトリの情報をアクセスするように webMethods Integration Server を設定することもできます。

- セントラルユーザ管理
- LDAP (Lightweight Directory Access Protocol)

この章では、ユーザとグループの情報を内部で定義している場合に Integration Server がどのように動作するかについてのみ説明します。Integration Server での外部ディレクトリ使用の詳細については、675 ページの「セントラルユーザディレクトリまたは LDAP の設定」を参照してください。

ユーザとグループの目的

Integration Server では、ユーザとグループの情報によってクライアントを認証し、クライアントがアクセスを許可されているサーバリソースを判断します。

サーバが基本的な認証 (ユーザ名およびパスワード) をクライアントの認証に使用している場合、ユーザアカウントに定義されたユーザ名およびパスワードを使用して、クライアントが提示するクレデンシャルの妥当性検査を行います。

クライアントが基本的な認証またはクライアントの認証によって認証された後、サーバはグループメンバーシップを使用して、クライアントが Integration Server Administrator の使用やサービスの呼び出しなどの要求アクションに対して認証されているかどうかを確認します。

サーバのリソースへのアクセスは、グループレベルで制御されます。ユーザとグループを設定することによって、次に示す項目を実行できるユーザを指定できます。

- **サーバの設定および管理**(Administrator 特権を付与されている) Administrators グループのメンバーであるユーザのみが Integration Server Administrator にアクセスできます。
- **サーバ上に存在するサービスの作成、変更、および削除**(Developer 特権を付与されている) Developers グループのメンバーであるユーザのみが Software AG Designer からサーバに接続できます。
- **サーバ上に存在するサービスおよびファイルへのアクセス**サービスおよびファイルへのアクセスは、グループレベルで保護されます。

ユーザアカウントの定義

Integration Server でユーザアカウントを作成する場合は、ユーザ名、パスワードおよびグループメンバーシップを指定します。

- **ユーザ名**ユーザ名はクライアントを識別する一意の名前です。ユーザ名には、実際の人名を表すユーザ名 (John D. Smith を表す「JDSmith」など) や用途、職務または所属組織などを表すユーザ名を指定できます。たとえば、「MktgPurchAgent」、「MktgTimeKeeper」など特定職務の一般名称を設定できます。

- **パスワード**パスワードは、ユーザ名に関連付ける任意のストリングです。サーバは、有効なユーザ名を入力したクライアントを認証するときにパスワードを使用します。認証の詳細については、561 ページの「JAAS を使用した認証のカスタマイズ」を参照してください。

  パスワードは、サーバ、サーバ管理者およびユーザアカウントの所有者のみが共有する暗証コードです。その目的は、要求が正当なユーザから送信されていることをサーバに保証することです。ユーザ名にパスワードを割り当てたり、既存アカウントのパスワードを変更したりできるのは管理者のみです。なお、セキュリティの強化のため、パスワードはハッシュされてから保存されます。

  注:
  また、Integration Server は、認証オプションとして、パスワードダイジェストを提供します。

- **グループメンバーシップ**グループメンバーシップは、ユーザが属するグループを識別します。サーバのリソースへのアクセスは、グループレベルで制御されます。

  Integration Server Administrator を使用してサーバの設定および管理ができるのは、Administrators グループのメンバーであるユーザだけです。Integration Server Administrator へのアクセスの制御方法の詳細については、1292 ページの「FIPS 140-2 準拠」を参照してください。

  Software AG Designer からサーバに接続して、サービスの作成、変更および削除ができるのは、Developers グループのメンバーであるユーザだけです。詳細については、107 ページの「Developer ユーザの追加」を参照してください。

  サーバは、サービスおよびファイルへのアクセスを ACL を使用して保護します。ACL を設定して、リソースへのアクセスが許可されているグループと許可されていないグループを識別します。サービスおよびファイルの保護の詳細については、534 ページの「ACL によるリソースへのアクセス制御」を参照してください。

事前定義済みのユーザアカウント

サーバには、次に示すユーザアカウントが事前に定義されています。

ユーザ	グループ	説明
Administrator	Everybody Administrators Replicators	Administrator 特権を持つユーザアカウント。Administrator ユーザアカウントを使用すると、Integration Server Administrator にアクセスしてサーバを設定および管理できます。
Default	Everybody Anonymous	サーバは、クライアントがユーザ ID を入力しない場合にデフォルトのユーザに定義された情報を使用します。

ユーザ	グループ	説明
Developer	Everybody Developers	サーバに Software AG Designer から接続し、そのサーバ上に存在するサービスの作成、変更および削除ができるユーザ。 Developer ユーザアカウントはデフォルトで無効になっています。Developer ユーザアカウントを有効にする前に、最初に Developer ユーザアカウントのパスワードを設定する必要があります。
Replicator	Everybody Replicators	パッケージ複製中にサーバが使用するユーザアカウント。パッケージの複製の詳細については、716 ページの「サーバ間でのパッケージのコピー」を参照してください。 Replicator ユーザアカウントはデフォルトで無効になっています。Replicator ユーザアカウントを有効にする前に、Replicator ユーザアカウントのパスワードを設定する必要があります。

重要:
Software AG では、事前定義済みアカウントを使用する代わりに各管理者および開発者の個々のアカウントを作成する最良の方法をお勧めします。個々のアカウントの作成後、事前定義済みアカウントを無効にします。

ユーザアカウントの追加

ユーザ用にユーザアカウントを追加するには、以下の手順に従います。

ユーザアカウントをサーバに追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**ユーザの追加と削除**] をクリックします。
4. ページの [**ユーザの作成**] セクションで、次の情報を指定します。

パラメータ	指定内容
[ユーザ名]	文字、数字または記号を組み合わせた一意のユーザ名。行ごとにユーザ名を 1 つ指定できます。行を分割するには Enter キーを押します。 **重要:**

パラメータ	指定内容
	ユーザ名は大文字小文字を区別します。ユーザアカウントを作成するときには、クライアントが入力する文字列を正確に入力してください。
	注: ストリング「SAMLart」は Integration Server での予約語です。この語を含んだ Integration Server ユーザ名は作成しないでください。
[パスワード]	文字、数字または記号を組み合わせたパスワード。 パスワードは必要で、NULL 値または空の文字列にできません。
	重要: パスワードは大文字小文字を区別します。パスワードの値には、クライアントが入力する文字列を正確に入力してください。
	必ず、推測しにくいパスワードを選択してください。たとえば、大文字、小文字、数字および特殊文字を組み合わせて使用します。名前、電話番号、保険証番号、車のプレートナンバーなど、入手しやすい情報は使用しないでください。
[パスワードの再入力]	正しく入力したことを確認するために、同じパスワードを再度入力します。

5. [**Digest 認証の許可**] をオンにすると、認証オプションとしてパスワードダイジェストが使用されます。

6. [**ユーザの作成**] をクリックします。

ユーザアカウントの削除

ユーザアカウントが必要なくなった場合、アカウントを削除できます。ただし、ユーザアカウントを削除する前に、トリガーまたはユーザタスクの実行ユーザ、クライアント認証のユーザマッピング、リモートサーバエイリアスなどの送信接続設定など、Integration Server でアカウントが使用されるすべての場所を更新してください。ユーザアカウントが使用されている可能性がある場所の詳細リストについては、1300 ページの「ユーザアカウントへの参照の削除」を参照してください。

ユーザアカウントを削除する場合は、以下の点に留意してください。

- ユーザを削除すると、そのユーザは Integration Server によって割り当て先のすべてのグループのメンバーリストから自動的に削除されます。
- 組み込みユーザアカウントである Administrator、Default、Developer および Replicator は削除できません。
- ユーザアカウントを削除しても、ユーザアカウントデータまたはログファイルなどに含まれる他の識別データがすべて削除されるわけではありません。ログファイルからユーザデータ

を削除する方法については、1299 ページの「Integration Server からのユーザデータの削除」を参照してください。

ユーザアカウントをサーバから削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**ユーザの追加と削除**] をクリックします。
4. ページの [**ユーザの削除**] セクションで、削除するユーザアカウントのユーザ名を選択します。
5. [**ユーザの削除**] をクリックします。ユーザアカウントの削除を確認するプロンプトが表示されます。ユーザアカウントを削除するには、[**OK**] をクリックします。

Administrator ユーザの追加

ユーザが、Administrator グループまたは Administrator ACL の許可リストに追加されているその他のグループに所属している場合は、そのユーザには Administrator 特権が付与されます。ユーザが Administrator 特権を持っているかどうかを判断するために、サーバはユーザの認証を行ってそのユーザの名前を取得します(サーバがユーザ名を識別する方法の詳細については、561 ページの「JAAS を使用した認証のカスタマイズ」を参照してください)。ユーザ名の識別後、そのユーザが Administrators ACL によってアクセスが許可されたグループに属し、拒否されたグループに属していないかどうかが確認されます。所属グループが許可されたグループであれば、サーバは Integration Server Administrator へのアクセスを許可します。

ユーザに Administrator 特権を付与するには、Administrators グループ、または Administrators ACL の許可リストに含まれるグループにそのユーザを割り当てる必要があります。さらに、そのユーザが Administrators ACL によってアクセスが拒否されたグループのメンバーでないことも確認する必要があります。

重要:
ユーザに Administrator 特権を付与するには、そのユーザが既に Integration Server のユーザアカウントを持っている必要があります。持っていない場合は、まずユーザアカウントを作成してから、以下の手順に従います。

ユーザに Administrator 特権を付与するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。

   [**ローカルユーザ管理**] のページ右側の [**グループ**] 領域には 2 つのリストがあります。[**所属ユーザ**] は、選択したグループに現在所属するユーザのリストです。[**候補ユーザ**] は、選択したグループに現在所属していないユーザのリストです。

3. ページの [**グループ**] 領域の [**グループを選択**] リストから [**Administrators**] を選択します。

4. [**候補ユーザ**] リストで、Administrator 特権を付与するユーザを選択して (複数選択可) ハイライトします。

   現在選択しているユーザを解除せずにさらにユーザを追加するには、Ctrl キーを押しながら、追加するユーザをクリックします。ユーザの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

5. グループに追加するユーザをすべて選択したら、 をクリックします。選択したユーザが [**所属ユーザ**] リストに移動されます。

6. [**変更内容の保存**] をクリックします。

Developer ユーザの追加

Developer は Software AG Designer を使用して、サーバ上のパッケージ、フォルダ、サービスおよびその他のエレメントを表示、作成、変更、削除することができます。Designer からアクセスする際には、ユーザの Developer 特権の有無が確認されます。

ユーザが、Developer グループまたは Developer ACL の許可リストに追加されているその他のグループに所属している場合は、そのユーザには Developer 特権が付与されます。ユーザが Developer 特権を持っているかどうかを判断するために、サーバはユーザの認証を行ってそのユーザの名前を取得します(サーバがユーザ名を識別する方法の詳細については、561 ページの「JAAS を使用した認証のカスタマイズ」を参照してください)。ユーザ名の識別後、そのユーザが Developers ACL によってアクセスが許可されたグループに属し、拒否されたグループに属していないかどうかが確認されます。所属グループが許可されたグループであれば、Designer とサーバ間の接続の確立が許可されます。

ユーザに Developer 特権を付与するには、Developers グループまたは Developers ACL の許可リストに含まれるグループに、そのユーザを割り当てる必要があります。さらに、そのユーザが Developer ACL によってアクセスが拒否されたグループのメンバーでないことも確認する必要があります。

重要:
リスト ACL、読み取り ACL、書き込み ACL は、予期しない干渉や破損からエレメントを保護するメカニズムです。開発者は意図的にこのメカニズムを回避することができます。悪意による侵害から保護する場合は、ACL に依存しないようにしてください。

重要:
ユーザに Developer 特権を付与するには、そのユーザが既に Integration Server のユーザアカウントを持っている必要があります。持っていない場合は、まずユーザアカウントを作成してから、以下の手順に従います。

ユーザに Developer 特権を付与するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [ユーザ管理] に移動します。

   [**ローカルユーザ管理**] のページ右側の [**グループ**] 領域には 2 つのリストがあります。[**所属ユーザ**] は、選択したグループに現在所属するユーザのリストです。[**候補ユーザ**] は、選択したグループに現在所属していないユーザのリストです。

3. ページの [**グループ**] 領域の [**グループを選択**] リストから [**Developers**] を選択します。

4. [**候補ユーザ**] リストで、Developer 特権を付与するユーザ選択して (複数選択可) ハイライトします。

   現在選択しているユーザを解除せずにさらにユーザを追加するには、Ctrl キーを押しながら、追加するユーザをクリックします。ユーザの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

5. グループに追加するユーザをすべて選択したら、 をクリックします。選択したユーザが [**所属ユーザ**] リストに移動されます。

6. [**変更内容の保存**] をクリックします。

パスワードの変更

自分のユーザアカウントまたは他人のユーザアカウントのパスワードを変更することができます。パスワードを変更する場合は、以下の点に留意してください。

- 企業ファイアウォールの外から SSL を使用せずに Integration Server に接続している場合は、パスワードを変更しないでください。
- パスワードは値または空の文字列にできません。
- 必ず、推測しにくいパスワードを選択してください。たとえば、大文字、小文字、数字および特殊文字を組み合わせて使用します。システムのセキュリティに関わることなので、名前、電話番号、保険証番号、車のプレートナンバーなど、入手しやすい情報は使用しないでください。
- Integration Server Administrator または Software AG Designer は、外部ディレクトリに保存されたユーザまたはグループの管理には使用できません。また、外部ディレクトリに保存されたユーザのパスワードも変更できません。

ユーザのパスワードを変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [ユーザ管理] に移動します。

3. ページの [**ユーザ**] セクションで、パスワードを変更するユーザを選択して、[**パスワードの変更**] をクリックします。

4. 次の情報を入力します。

パラメータ	指定内容
[新しいパスワード]	文字、数字または記号を組み合わせた新しいパスワード。NULL 値または空の文字列は無効です。 **重要:** パスワードは大文字小文字を区別します。パスワードの値には、クライアントが入力する文字列を正確に入力してください。 必ず、推測しにくいパスワードを選択してください。たとえば、大文字、小文字、数字および特殊文字を組み合わせて使用します。名前、電話番号、保険証番号、車のプレートナンバーなど、入手しやすい情報は使用しないでください。
[パスワードの確認]	正しく入力したことを確認するために、同じパスワードを再度入力します。

5. [**Digest 認証の許可**] をオンにすると、認証オプションとしてパスワードダイジェストが使用されます。

6. [**パスワードの保存**] をクリックします。

パスワード要件と有効期限の設定

セキュリティ上の理由により、ユーザアカウントのパスワード要件を指定できます。Integration Server Administrator を使用すると、以下の内容を指定できます。

- パスワードの長さと構成の要件。詳細については、以下を参照してください。109 ページの「パスワードの要件の設定」
- パスワードの有効期限の設定。詳細については、以下を参照してください。111 ページの「パスワードの有効期限要件の設定」
- アカウントロックの設定。詳細については、以下を参照してください。113 ページの「アカウントロックの設定」

パスワードの要件の設定

セキュリティ上の理由により、Integration Server では Administrator ユーザと Administrator 以外のユーザのパスワードの長さと文字に制限が設定されています。Integration Server にはデフォルトのパスワード要件がありますが、Integration Server Administrator で変更することができます。

管理者以外のユーザのパスワードの長さや文字の要件を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [ユーザ管理] に移動します。

3. [**パスワード制限**] をクリックします。

4. [**パスワード制限の編集**] をクリックします。

5. 次のフィールドに情報を入力します。

フィールド	説明	デフォルト
[パスワードの変更を有効にする]	ユーザにパスワードの変更を許可するかどうか。 ユーザは Developer 特権を持っている必要があります。	はい
[パスワード強制モード]	Administrator ユーザはパスワード制限設定に影響を受けないパスワードを選択できるかどうか。 Administrator ユーザに対し、このプロパティが「`Strict`」に設定されている場合、Integration Server はパスワード制限を実行します。「`Lax`」に設定した場合、パスワード制限は実行されません。 Administrator 以外のユーザの場合、常に制限が実行されます。	`Lax`
[パスワードの長さの最小値]	パスワードの最小文字数 (アルファベット文字、数字および特殊文字の組み合わせ)。	8
[パスワードの長さの最大値]	パスワードの最大文字数 (アルファベット文字、数字および特殊文字の組み合わせ)。パスワードに設定できる最大文字数は 128 です。	64
[アルファベットの大文字数の最小値]	パスワードに最低限入れる必要がある大文字のアルファベットの数。	0
[アルファベットの大文字数の最小値]	パスワードに最低限入れる必要がある小文字のアルファベットの数。	0
[数字の数の最小値]	パスワードに最低限入れる必要がある数字の数。	0
[特殊文字数の最小値 (アルファベットおよび数字以外の文字)]	パスワードに最低限入れる必要がある特殊文字 (*、. 、? など) の数。 **注:** パスワードの先頭文字にアスタリスク (*) を使用することはできません。	0
[連続する同一文字数の最大値]	パスワードに入れることができる同一文字の連続使用の最大値。	3
[記録する古いパスワードの数 (1 ユーザにつき)]	1 人のユーザに対して Integration Server で保存する、以前設定されたパスワードの最大数 (現在のパ	0

フィールド	説明	デフォルト
	スワードを除く)。保存されたパスワードに一致するパスワードを選択することはできません。保存されるパスワードの最大数は 12 です。 **注:** このフィールドに新しい値を設定すると、保存直後に有効になります。このフィールドに設定された古い値に対応する、以前保存されたすべてのパスワードは削除され、回復できません。 たとえば、最初に管理者がこのフィールドの値を 5 に設定した場合、Integration Server は最後の 5 個のパスワードを保存します。管理者がこのフィールドの値を 3 に変更した場合、Integration Server は最新の 3 個のパスワードエントリのみ保持し、古い 2 個のエントリは削除します。同様に、管理者が 6 に変更した場合、Integration Server は 6 個の最新エントリを保持します。	

6. [パスワード設定の保存] をクリックします。

パスワードの有効期限要件の設定

セキュリティ上の理由により、Integration Server では管理者が Administrator ユーザと Administrator 以外のユーザのパスワードに対し、パスワードの有効期限要件を設定できます。Integration Server にはパスワード有効期限設定のデフォルトセットがありますが、Integration Server Administrator で変更することができます。Administrator ユーザは Integration Server Administrator の [パスワードの有効期限の設定] ページで指定した日数より前にパスワードを再設定することを知らせる電子メールを受信します。

パスワードの有効期限を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**パスワードの有効期限の設定**] をクリックします。
4. [**パスワードの有効期限の設定の編集**] をクリックします。
5. ユーザに Administrator 特権を付与するには、以下を実行します。

フィールド	説明
[適用するユーザ]	有効期間設定を適用できるユーザのリスト。 <- と -> ボタンを使用して、ユーザ名をこのリストの中または外に移動します。
[残りのユーザ]	有効期間設定を適用できないユーザのリスト。 <- と -> ボタンを使用して、ユーザ名をこのリストの中または外に移動します。

6. 次のフィールドに情報を入力します。

フィールド	説明	デフォルト
[有効]	パスワードの有効期限の設定を有効にするかどうか。このオプションは、デフォルトでは無効になっています。パスワードの有効期限の設定を有効にするには、[**有効**] を選択します。	
[有効期間]	変更しないと、パスワードが期限切れになる日数。値は 0 以外の整数にする必要があります。 **注:** このオプションを有効にすると、保存時点でユーザが有効期間より前に設定したパスワードは期限切れと見なされ、ユーザは再設定する必要があります。たとえば、パスワードを 10 日前に変更し、現時点で管理者が有効期間を 5 日に変更した場合、パスワードは期限切れになり、再設定が必要になります。	90
[有効期限を通知する電子メールアドレス]	ユーザのパスワードが期限切れ寸前であるか既に期限切れになったことを知らせる電子メール通知を Integration Server が送信する先の電子メールアドレスのリスト。フィールドを空白のままにした場合、Integration Server はリソース設定ページ ([設定] > [リソース]) の [電子メール通知] セクションの [内部用電子メール] フィールドで指定された電子メールアドレスを使用します。 **注:** Integration Server はリソース設定ページ ([設定] > [リソース]) の [電子メール通知] セクションに指定された SMTP サーバとポートの詳細を使用します。	
[有効期限の電子メール通知]	パスワードが期限切れになる何日前に、Integration Server がパスワードの再設定に関する電子メール	3

フィールド	説明	デフォルト
	通知を送信するかを指定します。ユーザがパスワードを更新するか有効期間を変更するまで、電子メールは毎日送信されます。値を 0 に設定すると、Integration Server は期限切れの前にパスワードに関する電子メール通知を送信しません。デフォルトは 3 です。 **注:** パスワードが期限切れになっても、ユーザがパスワードを再設定するまで、Integration Server は電子メール通知の送信を続行します。値を 0 に設定しても、期限切れのパスワードに関する電子メール通知は送信されます。	

7. [**変更内容の保存**] をクリックします。

アカウントロックの設定

ユーザが Integration Server へのログインを指定回数失敗した後も正しいパスワードを指定できなかった場合、セキュリティ上の理由により、ユーザアカウントをロックすることが重要です。ロックされたユーザアカウントは特定の期間ロックされた後、アンロックされます。Integration Server の管理者は Administrator ユーザと Administrator 以外のユーザに対してアカウントロックを設定できます。Integration Server Administrator を使用して、アカウントをロックするまでにユーザが行う試行数の値、およびロック間隔の期間を設定できます。また、Integration Server の管理者はロックされたユーザアカウントを手動でアンロックすることもできます。

アカウントのロックを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**アカウントロックの設定**] をクリックします。
4. [**アカウントロックの設定の編集**] をクリックします。
5. 次のフィールドに情報を入力します。

フィールド	説明	デフォルト
[有効]	アカウントのロックの設定を有効にするかどうか。アカウントのロックの設定を有効にするには、[**有効**] を選択します。	

フィールド	説明	デフォルト
[ログインの最大試行回数]	特定の期間 (分、時間、または日) 内で、アカウントのロック前に正しいパスワードを指定する試行数。	なし
[ロックアウトの経過時間]	アカウントがロックされている時間。	なし
[アカウントロックポリシーの適用先]	アカウントロック設定の適用先ユーザのリスト。 次のいずれかを指定します。 ■ [事前定義されたユーザ以外のすべてのユーザ] - アカウントロックルールが、事前定義済みのユーザアカウント (Administrator、Default、Developer、Replicator) を除くすべてのユーザアカウントに適用されることを示します。 ■ [すべてのユーザ] - アカウントロックルールがすべてのユーザアカウントに適用されることを示します。	[事前定義されたユーザ以外のすべてのユーザ]

6. [**変更内容の保存**] をクリックします。

ユーザアカウントのアンロック

Integration Server では指定されたロック期間後にユーザアカウントをアンロックします。ただし、Integration Server Administrator を使用すると、ユーザアカウントを手動でアンロックできます。

ユーザアカウントをアンロックするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**アカウントロックの設定**] をクリックします。
4. [**ロックされたユーザ**] をクリックします。
5. ロックされたユーザアカウントのリストから ✕ をクリックして、ユーザアカウントをアンロックします。
6. [**変更内容の保存**] をクリックします。

ユーザアカウントの無効化と有効化

ユーザアカウントを無効にする必要がある場合があります。よく知られているユーザ名を無効にすることによって、パスワードのクラック攻撃が困難になります。ユーザアカウントを無効にした場合、そのユーザ名でログインしようとすると、認証が失敗してログインが拒否されます。たとえば、休暇中の開発者のユーザアカウントや、取引特権が一時的に停止されているトレーディングパートナのアカウントを無効にすることができます。この場合、ユーザは削除されるのではなく、無効になるだけなので、パスワードの変更やアクセス許可のリセットをしなくても、アカウントを後で回復することができます。

展開する場合は、Administrator ユーザを無効にして、誰かがパスワードを推測してシステムへのアクセス権限を取得することがないようにする必要があります。Administrator ユーザを無効にする前に、まず SmithAdmin などの別のユーザを作成して、Administrators、Developers および Replicators グループに追加する必要があります。その後 Administrator ユーザを無効にします(起動サービス、シャットダウンサービスなど、Administrator ユーザとして実行する内部サーバ機能は、そのまま Administrator として実行できます)。この後、SmithAdmin ユーザを使用して Integration Server を管理することができます。

ユーザアカウントの無効化

ユーザアカウントを無効にするには、以下の手順に従います。

重要:
Administrator ユーザを無効にする際は、まず Administrator 特権を持つ別のユーザを定義して、サーバからロックアウトされないようにしてください。

ユーザアカウントを無効化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**ユーザの有効化と無効化**] をクリックします。
4. [**有効なユーザ**] リストで、無効にするユーザを選択して (複数選択可) ハイライトします。

   現在選択しているユーザを解除せずにさらにユーザを追加するには、Ctrl キーを押しながら、追加するユーザをクリックします。ユーザの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。
5. ページの [有効なユーザ] 領域の下にある をクリックします。

   選択したユーザがページの [**無効なユーザ**] 領域に移動されます。
6. [**変更内容の保存**] をクリックします。

ユーザアカウントの有効化

ユーザアカウントを有効にするには、以下の手順に従います。ユーザアカウントを有効化する必要が生じるのは、システム管理者が明示的にユーザを無効化した場合です。

注:
Developer および Replicator のユーザアカウントはデフォルトで無効になっています。有効化する前に、これらのユーザアカウントのパスワードを設定する必要があります。

ユーザアカウントを有効化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**ユーザの有効化と無効化**] をクリックします。
4. [**無効なユーザ**] リストで、有効にするユーザを選択して (複数選択可) ハイライトします。

   現在選択しているユーザを解除せずにさらにユーザを追加するには、Ctrl キーを押しながら、追加するユーザをクリックします。ユーザの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。
5. ページの [無効なユーザ] 領域の下にある をクリックします。

   選択したユーザがページの [**有効なユーザ**] 領域に移動されます。
6. [**変更内容の保存**] をクリックします。

グループの定義

グループとは、特権を共有するユーザの集合に名前を付けたものです。次のような特権があります。

- Administrator 特権
- Replicator 特権
- Developer 特権
- サービスを呼び出す特権
- サーバにファイルの提供を許可する特権

サービスを呼び出したりファイルにアクセスしたりする特権は、設定した ACL によって許可または拒否されます。管理者が ACL を作成するときには、サービスおよびファイルへのアクセスを許可するグループと、サービスおよびファイルへのアクセスを拒否するグループを指定します。

Administrator 特権、Replicator 特権および Developer 特権を付与するには、通常、ユーザをそれぞれ Administrators、Replicators または Developers グループに追加します。または、新規グループを作成して、そのグループを Administrators ACL、Replicators ACL または Developers ACL の許可リストに追加します。

同じ特権を持つユーザをまとめてグループを作成します。グループ定義を作成する場合は、グループ名とそのメンバーの名前を指定します。

- **グループ名**グループ名はグループを識別する一意の名前です。名前は、部門を定義する名前 (Marketing) や職務を定義する名前 (Programmers) など、任意の名前を付けることができます。
- **メンバー**グループのメンバーであるユーザ名のリストです。

事前定義済みのグループ

Integration Server は、次に示す事前定義済みのグループと共にインストールされます。

グループ名	メンバー	説明
Administrators	Administrator	このグループは、Administrator 特権があるユーザを識別します。サーバを設定、管理するには、この Administrator 特権が必要です。 **重要:** このグループのメンバーには、Integration Server の設定を変更する強力な権限があります。したがって、このグループには、この特権を慎重に行使することに関して信頼できる人を割り当ててください。
Anonymous	Default	このグループは、ユーザ ID を指定しなかったユーザです。
Developers	Developer	このグループは、Developer 特権があるユーザを識別します。Designer からサーバに接続するには、この Developer 特権が必要です。 **重要:** このグループのメンバーには、Integration Server の設定を変更する強力な権限があります。したがって、このグループには、この特権を慎重に行使することに関して信頼できる人を割り当ててください。
Everybody	Administrator Default Developer Replicator	すべてのユーザはこのグループのメンバーです。新規ユーザはすべて自動的に Everybody グループに追加されます。

グループ名	メンバー	説明
Replicators	Administrator Replicator	このグループは、Replicator 特権があるユーザを識別します。Replicators グループは、メンバーにパッケージの複製を実行する権限を付与します(デフォルトで、サーバはパッケージの複製に Replicators グループのメンバーを使用します)。 ユーザは、パッケージの複製を実行するために Replicators グループのメンバーになる必要はありません。ユーザが Replicators ACL に割り当てられているグループのメンバーである限り、そのユーザはパッケージの複製を実行できます。 パッケージの複製の詳細については、716 ページの「サーバ間でのパッケージのコピー」を参照してください。 このグループのメンバーには、Integration Server の設定を変更する強力な権限があります。したがって、このグループには、この特権を慎重に行使することに関して信頼できる人を割り当ててください。

グループの追加

グループを追加するには、以下の手順に従います。

新規グループをサーバに追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**グループの追加と削除**] をクリックします。
4. ページの [**グループの作成**] 領域で、文字、数字または記号を組み合わせた一意のグループ名を入力します。

   重要:
   グループ名には、空白およびカンマ (,)、引用符 (' または ")、バックスラッシュ (\)、スラッシュ (/) などの特殊文字は使用できません。

   一度に複数のグループを追加するには、1 行に 1 グループずつ複数の行を指定します。行を分割するには Enter キーを押します。
5. [**グループの作成**] をクリックします。

グループへのユーザの追加

グループにユーザを追加するには、以下の手順に従います。

注:
Everybody グループのメンバーシップを変更することはできません。

グループにユーザを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。

   次のページが表示されます。

   ページ右側の [**グループ**] 領域には 2 つのリストがあります。[**所属ユーザ**] は、現在グループに所属しているユーザのリストです。[**候補ユーザ**] は、現在はグループに所属していないユーザのリストです。
3. [**グループ**] の [**グループを選択**] リストで、ユーザを追加するグループを選択します。
4. [**候補ユーザ**] リストで、グループに追加するユーザを選択して (複数選択可) ハイライトします。

   現在選択しているユーザを解除せずにさらにユーザを追加するには、Ctrl キーを押しながら、追加するユーザをクリックします。ユーザの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。
5. グループに追加するユーザをすべて選択したら、 をクリックします。選択したユーザが [**所属ユーザ**] リストに移動されます。
6. [**変更内容の保存**] をクリックします。

グループからユーザを削除

グループからユーザを削除するには、以下の手順に従います。

注:
Everybody グループのメンバーシップを変更することはできません。

グループからユーザを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。

   次のページが表示されます。

   ページ右側の [**グループ**] 領域には 2 つのリストがあります。[**所属ユーザ**] は、現在グループに所属しているユーザのリストです。[**候補ユーザ**] は、現在はグループに所属していないユーザのリストです。

3. [**グループ**] の [**グループを選択**] リストで、ユーザを削除するグループを選択します。
4. ページの [**所属ユーザ**] 領域で、グループから削除するユーザを選択して (複数選択可) ハイライトします。

   現在選択しているユーザを解除せずにさらにユーザを追加するには、Ctrl キーを押しながら、追加するユーザをクリックします。ユーザの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

5. ページの [所属ユーザ] 領域の下にある -> をクリックします。選択したユーザがページの [**候補ユーザ**] 領域に移動されます。
6. [**変更内容の保存**] をクリックします。

グループメンバーシップの表示

メンバーまたはグループを表示したり、グループ内のメンバーを変更したりするには、以下の手順に従います。

グループのメンバーシップを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [ユーザ管理] に移動します。

   次のページが表示されます。

   ページ右側の [**グループ**] 領域には 2 つのリストがあります。[**所属ユーザ**] は、選択したグループに現在所属しているユーザのリストです。[**候補ユーザ**] は、選択したグループに現在所属していないユーザのリストです。

3. [**グループ**] の [**グループを選択**] リストで、メンバーシップを表示するグループを選択します。

4. そのグループのメンバーが [**所属ユーザ**] リストに表示されます。

グループの削除

不要になったグループを削除するには、以下の手順に従います。

注:
Administrators、Developers、Replicators、Anonymous および Everybody グループは、いずれも削除できません。

グループをサーバから削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [ユーザ管理] に移動します。

3. [**グループの追加と削除**] をクリックします。

4. ページの [**グループの削除**] 領域で、削除するグループを選択します。

5. [**グループの削除**] をクリックします。

6. [**変更内容の保存**] をクリックします。

8 サーバの設定

- ライセンス情報の表示および変更 124
- サーバスレッドプールの管理 127
- サーバセッションの管理 128
- 送信 HTTP の設定 131
- リモート Integration Server に対するエイリアスの設定 133
- 送信要求に対するサードパーティのプロキシサーバの指定 137
- Integration Server によるログ、状態、その他の情報の保存場所の設定 145
- 組み込みデータベースから外部 RDBMS への切り替え 146
- 拡張設定の使い方 146
- 文字エンコーディングの指定 147
- JVM の設定 148
- Integration Server の JDK または JRE の指定 148
- Integration Server が使用する JVM ヒープサイズの変更 149
- Integration Server のアセット情報のパブリッシュおよび取り消し 150
- リモートクライアント JMX 監視のポート設定 152
- 起動中のデバッグ接続を受け入れるためのIntegration Server 設定 153
- Integration Server での CORS の使用 154

ライセンス情報の表示および変更

webMethods Integration Server を購入すると、特定の機能について所定数の同時ユーザ (同時セッション) で使用できるライセンスが発行されます。

このライセンスは、売買契約で指定された一定期間を過ぎると失効します。

ライセンスキー

Integration Server をインストールする前に、ライセンスキーファイルが提供されるので、Integration Server を実行するマシンのファイルシステムにそのライセンスキーファイルを配置します。このファイルには、ライセンスに関連付けられた特殊コードであるライセンスキーが含まれています。

Integration Server のインストール時に、セットアッププログラムによってこのファイルの名前および場所を入力するように要求されます。続いて、セットアッププログラムはこのファイルを licenseKey.xml という名前で *Integration Server_directory* \instances*instance_name*\config ディレクトリにコピーします。このファイルを誤って削除してしまうと、Integration Server はデモモードに戻ります。デモモードでライセンスが許可されているセッションは 2 つのみです。サーバは、開始後 30 分を経過すると自動的にシャットダウンします。

ライセンス情報の表示

Integration Server のライセンス情報を表示するかライセンスキーを変更するには、Integration Server Administrator の [**ライセンス**] ページを使用します。

ライセンス情報を表示するには

1. Integration Server Administrator を開きます。
2. [**サーバ**] > [ライセンス] に移動します。

   Integration Server によって機能リストが表示され、ライセンスで使用が許可されている各機能の横にチェックマークが表示されます。
3. ライセンス情報の詳細を表示するには、[**ライセンスの詳細**] をクリックします。

   Integration Server によって、現行のライセンスキー、ライセンスで使用が許可されている機能のリスト、サーバ上で実行できる同時セッションの最大数、パートナおよび Trading Networks の情報などの詳細情報が表示されます。

ライセンス情報の変更

ライセンスの有効期限が切れたか、またはライセンスを変更して別の機能を追加する場合には、Integration Server Administrator を使用して Integration Server のライセンスファイルを変更します。

Docker コンテナで実行中の Integration Server イメージに使用されているライセンスファイルの変更の詳細については、1283 ページの「環境変数」 の SAG_IS_LICENSE_FILE ENV 変数の説明を参照してください。

注:
Integration Server Administrator のライセンス情報を変更する前に、Software AG から新しいライセンスキーを取得して、Integration Server を実行するマシンのファイルシステムにコピーしておく必要があります。

Integration Server. に使用するライセンスキーを変更するには、以下の手順に従います。

ライセンスキーを変更するには

1. Integration Server Administrator を開きます。
2. [**サーバ**] > [ライセンス] に移動します。
3. [**ライセンスの詳細**] をクリックします。
4. [**ライセンスの詳細の編集**] をクリックします。

   [**ライセンスの詳細の編集**] ページが表示されます。
5. [**Integration Serverライセンスファイル**] フィールドに、Software AG から取得したライセンスキーファイルのパス名を入力します。
6. [**変更内容の保存**] をクリックします。

   Integration Server はライセンスキーファイルの内容を *Integration Server_directory* \instances *instance_name*\config\licenseKey.xml にコピーし、その名前を示すように [**Integration Serverライセンスファイル**] フィールドを更新します。

   注:
   [**変更内容の保存**] をクリックすると、Integration Server によって有効期限が自動的に更新されます。Integration Server を再起動する必要はありません。

更新のお知らせ

ライセンス有効期限が切れる約 30 日前に、ライセンスの更新時期が近づいていることを知らせる電子メールメッセージが、Integration Server によって管理メッセージの受取人宛に自動的に送信されます。また、Integration Server Administrator のすべてのページの上部に次のメッセージが表示されます。

```
ライセンスキーは約 xxx 日後に期限切れになります … 新しいキーについては Software AG にお問い合わせください。
```

キーの更新

新しいキーを取得する場合、または現在のライセンスを更新する場合は、Software AG のカスタマケアまでご連絡ください。

ライセンスされた機能の追加

使用できる機能は Software AG との契約に基づきます。契約に機能を追加するには、Software AG のカスタマケアまでご連絡ください。

ライセンスされているセッション数

指定するライセンスにより、所定数のユーザが Integration Server で同時にセッションを持つことができます。開発者が Software AG Designer からサーバに接続するか、または IS クライアントからサーバに接続してサービスを実行すると、Integration Server によってセッションが作成されます。使用中のセッションが最大数になっているときにユーザがサーバへのアクセスを試みると、その要求は拒否され、次のようなエラーメッセージが返されます。

```
サーバがクライアント制限に到達しました。
```

現在アクティブなセッションの数、およびライセンスされているセッション数の上限は Integration Server Administrator の [**統計情報**] ページで表示できます。この値は永続的にライセンスキーと関連付けられ、 新しいライセンスを取得しない限り 変更できません。

Administrator 以外のユーザ (Administrators グループの一員でないユーザ) 1 人がサーバに接続するたびに、ライセンスされたセッションが 1 つ使用中ということになります。セッションは、タイムアウト (サーバの [セッションのタイムアウト] 設定) になるか、または要求者が wm.server:disconnect サービスを呼び出してセッションを終了するまで続きます。

ユーザがステートレスサービスを呼び出したときに、そのユーザ用のセッションがまだ存在していないと、セッションが自動的に作成されます。この場合、ユーザが Administrator でないと、セッションが 1 つ使用中になります。サービスが完了するとセッションは削除され、使用中のライセンス済みセッションの数が減少します。

注:
Integration Server で複数の要求を同時に受信したときに、これらの要求を処理するリソースがないと、サーバのパフォーマンスが低下する可能性があります。サーバに同時に存在するステートフルセッションの上限を調整することで、パフォーマンスを調整できます。詳細については、128 ページの「サーバセッションの管理」を参照してください。

アクティブなセッションの表示

アクティブなセッションの数を表示したり、ライセンスされているセッション数の上限を確認したりするには、以下の手順に従います。

アクティブなセッションの数を表示したり、ライセンスされているセッション数の上限を確認したりするには

1. Integration Server Administrator を開きます。

2. [**サーバ**] > [統計情報] に移動します。

   [**すべてのセッション**] フィールドに現在アクティブなセッションの数が表示されます。また、ライセンスされているセッション数の上限は [**ライセンスされているセッション**] フィールドに表示されます。

   アクティブセッションの詳細については、[**すべてのセッション**] フィールドの数をクリックしてください。

サーバスレッドプールの管理

サーバの最大スレッド数および最小スレッド数を調整することによって、サーバのパフォーマンスを向上させることができます。スレッドは、サービスの実行、メッセージングプロバイダからのドキュメント抽出、およびトリガーの実行に使用されます。サーバ起動時のスレッドプールには、初期値として最小スレッド数が設定されます。最大スレッド数に達するまで、サーバは必要に応じてスレッドをプールに追加します。最大スレッド数に達した場合、サーバは、現在のプロセスが完了してスレッドがプールに戻されるまで待機してから、次のプロセスを開始します。

また、使用可能なスレッドの警告レベルを設定することもできます。使用可能なスレッドの割合が警告レベル以下になると、ジャーナルログメッセージが生成され、使用可能なスレッドが減少していることを示す警告が通知されます。使用可能なスレッドの数がしきい値を超過すると、別のジャーナルログメッセージが生成されます。

サーバで実行中のシステムスレッドを表示するには、[**サーバ**] > [統計情報] > [システムスレッド] ページに移動します。詳細については、755 ページの「スレッドのキャンセルまたは強制終了」を参照してください。

サーバスレッドプールを設定するには

1. Integration Server Administrator を開きます。

2. [**設定**] > [リソース] に移動します。

3. [**リソースの設定の編集**] をクリックします。

4. [**サーバスレッドプール**] で、サーバスレッドプールの設定を以下のように更新します。

パラメータ	指定内容
[**最大スレッド**]	サーバスレッドプールで保持されるスレッドの最大数。この最大スレッド数に達した場合、サーバは、現在のプロセスが完了してスレッドがプールに戻されるまで待機してから、次のプロセスを実行します。デフォルトは 75 です。
[**最小スレッド**]	サーバスレッドプールで保持されるスレッドの最小数。サーバ起動時のスレッドプールには、初期値としてここで指定された最小スレッド数が設定されます。スレッドは、[**最大スレッド数**]

パラメータ	指定内容
	フィールドで指定した最大数に達するまで、必要に応じてプールに追加されます。デフォルトは 10 です。
[使用可能な警告しきい値]	使用可能なスレッドが不足していることを示す警告を生成するしきい値。使用可能なサーバスレッドの割合がこの値に達すると、次のようなジャーナルログメッセージが生成され、現在使用可能なスレッドの割合が示されます。 `使用可能なスレッド警告しきい値を超過しました。` デフォルトは 15% です。 割合を入力して変更内容を保存すると、スレッドの数が自動的に計算され、 指定した割合の横に表示されます。 **ヒント:** 使用可能なスレッドの割合が警告レベル以下になったときには、webMethods messaging trigger に関してサーバで受信して処理するドキュメントの数を減らすことができます。詳細については、879 ページの「webMethods Messaging Trigger の管理」を参照してください。
[スケジューラスレッドのスロットル]	スケジューラ機能が使用できるサーバスレッドの割合。デフォルトは 75% です。

5. [**変更内容の保存**] をクリックします。

サーバセッションの管理

Integration Server では、接続するリモートクライアントごとに新しいセッションが開始されます。セッションが長期間アイドル状態または非アクティブ状態になった場合や、大量のステートフルセッションが同時に作成された場合、サーバのパフォーマンスに影響を与えることがあります。

Integration Server には、セッション管理に使用できるさまざまな制御が用意されています。具体的には次の操作が可能です。

- アイドルセッションがアクティブ状態を維持できる時間 (分数) の制限
- サーバで同時に作成可能なステートフルセッション数の制限
- 使用可能なステートフルセッションに対する警告レベルの設定

セッションのタイムアウト制限の設定

開始されたセッションは、クライアントアプリケーションからサーバに切断命令 (直ちに接続を強制終了) が明確に発行されるか、非アクティブ状態が続いた結果としてセッションがタイムアウトするかの、どちらかの状況になるまでアクティブ状態を維持します。

セッションが長時間アイドル状態となっている場合は、通常、クライアントが非アクティブ状態になっているか、またはクライアントとサーバ間の接続が切断されています。サーバは、非アクティブセッションを常に監視し、指定した時間を超えてアイドル状態が続いた場合はセッションを終了します。アイドル状態のセッションを消去するためのステップがサーバで実行されていない場合、セッションは無制限にアクティブ状態を維持することになり、貴重なサーバリソースの浪費につながります。

セッションのタイムアウト制限を設定するには

1. Integration Server Administrator を開きます。
2. [**設定**] > [リソース] に移動します。
3. [**リソースの設定の編集**] をクリックします。
4. [**セッション**] の [**セッションのタイムアウト**] フィールドで、アイドルセッションがアクティブ状態を維持できる最大分数 (つまり、アイドルセッションを終了するまでのサーバの待機時間) を入力します。

   [**セッションのタイムアウト**] パラメータを適切に設定するには、サーバを使用するクライアントについて精通しておく必要があります。クライアントで使用するプログラムがすべて Java の場合は、タイムアウト値の設定を通常 6～7 分に短縮することができます。この設定を調整して、使用サイトに適した値を見つけてください。サーバのデフォルトのタイムアウト制限は 10 分です。この値は、ほとんどのサイトに適しています。ただし、クライアントで要求と要求の間に常に大きな遅れ (10 分を超える) が出る場合は、この値を大きくしてください。
5. [**変更内容の保存**] をクリックします。

ステートフルセッションの制限の設定

Integration Server では、接続するリモートクライアントごとに新しいセッションが開始されます。このため、サーバで複数の要求を同時に受信したときに、これらの要求を処理できるリソースがないと、問題が発生する可能性があります。

同時セッションの許可数はライセンスで指定されています。ただし、Integration Server Administrator の [**リソース**] ページを使用してステートフルセッションの制限を設定することで、パフォーマンスを調整できます。ステートフルセッションの制限を設定しているときに、同時に存在するステートフルセッションの数がその制限を超えると、サーバは新しい要求を拒否して、ユーザにエラーメッセージを返します。

また、使用可能なステートフルセッションの警告レベルを設定することもできます。使用可能なステートフルセッションの割合が警告レベル以下になると、使用しているステートフルセッションと使用可能なステートフルセッションについて警告するメッセージがサーバログに生成されます。

ステートフルセッションの制限を設定するには

1. Integration Server Administrator を開きます。
2. [**設定**] > [リソース] に移動します。
3. [**リソースの設定の編集**] をクリックします。
4. [**セッション**] で、以下のようにサーバセッション設定を更新します。

パラメータ	指定内容
[ステートフルセッションの制限を有効にする]	Integration Server で同時に存在するステートフルセッションの数を制限するかどうか。ステートフルセッションの制限を有効にする場合は [**はい**]、有効にしない場合は [**いいえ**] を選択します。 有効にした場合、サーバ上に同時に存在できるステーフルセッションの数は [**ステートフルセッションの最大数**] パラメータで定義します。 Integration Server Administrator の [**サーバ**] > [統計情報] ページでステートフルセッションの統計情報を表示できます。無効にした場合も、ステートフルセッションの統計情報は収集され、[**サーバ**] > [統計情報] ページに表示されます。
[ステートフルセッションの最大数]	Integration Server に同時に存在できるステーフルセッションの最大数。最大数のステートフルセッションが使用中になっているときにユーザがサーバにアクセスしてステートフルサービスを実行しようとすると、その要求は拒否され、次のようなエラーメッセージが返されます。 `現在、サーバは新規リクエストを受け入れていません。` 値は正の整数にする必要があります。値が指定されていないか、または機能が無効になっている場合、同時セッションの最大数は Integration Server ライセンスファイルに 指定されているライセンス済みセッション数の上限で決まります。

[使用可能なステートフルセッション警告しきい値]	使用可能なステートフルセッションが不足していることを示す警告を Integration Server が生成するしきい値。使用可能なステートフルセッションの割合がこのプロパティの値以下になると、Integration Server によって以下のようなサーバログメッセージが生成されます。
	`同時に存在するステートフルセッションが`
	`最大数の {0}% に達しています。{1} 個のセッションが使用できます。`
	
	デフォルトは 25% です。

5. [**変更内容の保存**] をクリックします。

送信 HTTP の設定

送信 HTTP パラメータは、HTTP および HTTPS 送信要求 (クライアントのために Integration Server が発行する要求) の発行方法と処理方法を指定するために使用します。具体的には、このパラメータによってサーバの応答待機時間、失敗した要求の再試行回数などを制御します。

以下に説明するように、サーバの送信 HTTP 設定には、開発者が実行時にデフォルトを上書きできるものもあります。

パラメータ	説明
[ユーザエージェント]	Web ドキュメントの要求時にサーバから送信される HTTP ユーザエージェント要求ヘッダーで使用する値を指定します。ユーザエージェントヘッダーは、要求を発行しているブラウザの種類を Web サーバに通知するために使用します。Integration Server におけるユーザエージェントヘッダーは、Web サーバが推定する Integration Server のブラウザタイプを指します。このヘッダーを検査してクライアント機能を判断し、その結果によって応対方法を調整するという Web サーバもあります。 Integration Server のインストール時には、[**ユーザエージェント**] パラメータは Mozilla/4.0 [en] (WinNT; I) に設定されます。この値は必要に応じて変更できますが、設定する値は、サーバで実行するサービスの大半に通用するものにしてください。 開発者が、サーバで使用する**ユーザエージェント**の値を認識していることを確認してください。アプリケーションでそれぞれ異なる**ユーザエージェント**を必要とする場合は、アプリケーション要求に HTTP ユーザエージェントヘッダーを指定することで、実行時にサーバのデフォルトを上書きできます。
[リダイレクト回数の最大値]	Integration Server で許可される要求のリダイレクト (ターゲットサーバによって別の URL に自動的に送信される) 回数を指定します。要求が指定したリダイレクト回数を超えると、即座に Integration Server からクライアントへ I/O 例外が返されます。 Integration Server のインストール時には、[**リダイレクト回数の最大値**] は 5 に設定されます。アクセス先のターゲットが常にこの値よりも多く要求をリダイレクトする場合は、この値を大きくする必要があります (ターゲットをクラスタ化された環境で使用する場合に発生することがあります)。
[タイムアウト]	ターゲットサーバからの応答を待機する時間を指定します。指定した時間内に Integration Server で応答が受信されない場合は、要求の再試行が [**再試行回数**] パラメータで指定した回数になるまで実行されます。指定した再試行回数を超えると、例外が返されます。 Integration Server のインストール時には、[**タイムアウト**] パラメータは 300 秒 (5 分) に設定されます。ターゲットサーバからの応答を無期限に待機するように Integration Server を設定するには、このパラメータを「0」に設定します。

パラメータ	説明
	重要: [**タイムアウト**] パラメータを「0」に設定し、ターゲットサーバが要求に応答しない場合、スレッドプールの枯渇により、要求を行った Integration Server では新しい要求を処理できません。 watt.net.timeout サーバ設定パラメータを使用して、サーバが HTTP 要求の実行を待機する秒数を指定することもできます。詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。
[再試行回数]	タイムアウトした要求がサーバ ([**タイムアウト**] パラメータで指定した期間内に応答の受信がなかったサーバ) によって再発行される回数を指定します。 Integration Server のインストール時には、[**再試行回数**] は 0 に設定されます。つまり、指定した時間内にサーバで応答を受信しなかった場合は、サーバから自動的に例外が返されます。タイムアウトした要求の再試行 (再発行) をサーバに指定する場合は、[**再試行回数**] に 0 よりも大きい値を設定します。指定した回数になるまでサーバによって要求が再試行されます。 開発者が、サーバで使用する**再試行回数**の値を認識していることを確認してください。別の値を使用する必要がある場合は、明示的にサービスに**再試行回数**の値を割り当てることができます。

注:
上記の送信 HTTP の設定に加え、サーバ設定パラメータ watt.net.http.followRedirect を使用して、送信 HTTP 要求が 300～400 の応答コードを受信するときに Integration Server が自動的にリダイレクトに従うかどうかを制御できます。また、watt.net.http.redirect.performSSRFcheck を使用して、Integration Server がリダイレクト要求に従うときに、IP アドレスとプロトコルをチェックすることにより SSRF (Server-Side Request Forgery) を防ぐかどうかを指定することもできます。

送信 HTTP 設定の指定

送信 HTTP 設定を指定するには、以下の手順に従います。

送信 HTTP 設定を指定するには

1. Integration Server Administrator を開きます。
2. [**設定**] > [リソース] に移動します。
3. [**リソースの設定の編集**] をクリックします。
4. [**送信 HTTP の設定**] を以下のように指定します。

パラメータ	指定内容
[ユーザエージェント]	クライアントで値が指定されていない場合に、HTTP ユーザエージェントヘッダーに自動入力されるストリング。ストリングは、HTTP ヘッダーで表示されるとおりに、スペース、記号、および句読点なども正確に入力します。
[リダイレクト回数の最大値]	サーバからクライアントへ I/O 例外が返されるまでに実行できる要求のリダイレクト回数を示す整数。
[タイムアウト]	サービスを再試行するかまたはタイムアウトエラーをクライアントに返す前にターゲットサーバからの応答をサーバが待機する時間 (秒数) を示す整数。
[再試行回数]	クライアントに例外を返す前に、タイムアウトしたサービスをサーバが再試行する回数を示す整数。

5. [**変更内容の保存**] をクリックします。

リモート Integration Server に対するエイリアスの設定

リモートサーバに対してエイリアスを設定することができます。エイリアスを介しての通信は最適化されて、リモートサーバとのトランザクションが高速になります。

リモートエイリアスは、以下の場合に使用します。

- **ほかの Integration Server でサービスを呼び出す**エイリアスの作成後、pub.remote:invoke サービスおよび pub.remote.gd:* サービスを使用して、リモートサーバでサービスを呼び出すことができます。この際、リモートサーバはエイリアス別に設定します。
- **複数のクライアント認証を提示する**Integration Server は、すべてのサーバに単一のクライアントの認証を提示できるだけでなく、さまざまな SSL サーバに対して異なるクライアント認証を提示することもできます。さらに、Integration Server では他の組織によってこのために提供された認証を提示することもできますこのような SSL サーバに対してリモートエイリアスを設定することにより、さまざまな認証をより簡単に SSL サーバへ提示できるようになります。詳細については、を参照してください。
- **パッケージの複製を実行する**サブスクライバがパブリッシャーを指定してサブスクリプションを設定、またはパブリッシャーからパッケージをプルする場合は、パブリッシャーサーバをサブスクライバに対するリモートサーバとして定義する必要があります。サブスクライバサーバがどのような方法でパブリッシャーサーバに接続し、サブスクリプションを設定、またはパッケージをプルしているかは、エイリアスを参照するとわかります。詳細については、732 ページの「サブスクライバサーバ」を参照してください。

エイリアスの定義には、リモートサーバへ接続するためにサーバで必要とされる接続情報が含まれます。この定義によって、リモートサーバの IP アドレスやホスト名が識別され、リモートサーバへの接続に HTTP 接続または HTTPS 接続のいずれを使用するべきかがわかります。

リモートサーバに提供されるユーザ名およびパスワードもエイリアスを使用して識別されます。リモートサーバでは、このユーザ名とパスワードを使用してクライアントが認証され、さらに要求されたサービスを実行する権限がクライアントにあるかどうかが決定されます。

実際には、リモートサーバへのアクセスが可能になるのは、エイリアスを使用すると、ユーザがリモートサーバ上で認証ユーザとして振る舞うことができるためです。そのため、不正ユーザがリモートサーバのサービスへアクセスするのを阻止するために、エイリアスにはアクセスコントロール情報も含まれています。ユーザは、エイリアスの使用を保護する ACL を指定します。エイリアスの使用権限を付与されたクライアントによって要求が作成されると、リモートサーバ上でサービスが要求されます。エイリアスの使用権限を付与されていないクライアントによって要求が作成されると、サーバによってその要求は拒否され、リモートサーバ上のサービスは呼び出されません。

Integration Server には、ローカル Integration Server に接続する事前定義済みのリモートサーバエイリアス「local」があります。Monitor からトランザクションを再サブミットするとき、このエイリアスが使用されます。local エイリアスは Administrator ユーザアカウントを使用して、ローカル Integration Server に接続します。Monitor を使用して、失敗した要求を再サブミットする、または local リモートサーバエイリアスを他の目的に使用する計画がある場合、local エイリアスを編集し、Administrator パスワードを設定する必要があります。

リモート Integration Server のエイリアスの追加

リモート Integration Server に対するエイリアスを追加するには、以下の手順に従います。

リモートサーバに対するエイリアスを追加するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [リモートサーバ] に移動します。
3. [**リモートサーバエイリアスの作成**] をクリックします。
4. [**リモートサーバエイリアスのプロパティ**] を以下のように設定します。

パラメータ	指定内容
[エイリアス]	エイリアスとして使用する名前。リモートサーバに付ける名前は任意ですが、次の文字は使用できません。 #-&@^!%*:$./\\`;,~+=)(|}{|[><"。
[ホスト名またはIPアドレス]	エイリアスが作成されるリモートサーバのホスト名または IP アドレス (workstation5.webmethods.com など)。 **注:** このフィールドには、空白文字は使用できません。
[ポート番号]	サーバからの要求を受信待機するリモートサーバのポート番号 (5555 など)。

パラメータ	指定内容
ユーザ名	リモートサーバのユーザアカウントに対するユーザ名。このエイリアスを使用してサービスを呼び出すと、リモートサーバは認証およびアクセスコントロールにこのユーザアカウントを使用します。リモートサーバで呼び出すサービスにアクセス可能なユーザ名を指定します。
[パスワード]	ユーザアカウント内の [**ユーザ名**] に対応するパスワード。
[実行 ACL]	リモートサーバに対して設定されたこのエイリアスを使用できるのはどのユーザグループなのかを管理する ACL。ドロップダウンリストから ACL を選択します。デフォルトでは、内部 ACL によって管理されるグループのメンバーのみが、このエイリアスを使用できます。
[キープアライブ接続の最大値]	特定のターゲットエンドポイント用に確保しておく、クライアントのキープアライブ接続のデフォルト数を設定します。これが指定されていない場合は、5つのキープアライブ接続が保持されます。このフィールドでは、特定のリモートホストに保持されるクライアント接続の最大数を指定します。つまり、確立可能な接続の最大数ではなく、再使用のために保持される非アクティブなクライアント接続の数に対する制限です。
[キープアライブタイムアウト]	リモートサーバへのアイドル接続がサーバで維持される時間 (分) を指定します。この値により接続は、タイムアウトになるまで、再使用の可能性に備えて保持されます。指定したキープアライブタイムアウトの時間が経過すると、接続は終了され、HTTP リスナーによって新しい接続の作成が試行されます。
[SSL の使用]	サーバをリモートサーバに接続する際に SSL を使用するかどうかを指定します。SSL を使用する場合は [**はい**]、使用しない場合は [**いいえ**] を選択します。 **重要:** [**はい**] を選択する場合は、HTTPS 要求を受信待機するようにリモートサーバを設定する必要があります。
[キーストアエイリアス (オプション)]	Integration Server キーストアのユーザ指定のテキスト識別子。 エイリアスは鍵および関連する SSL 認証のリポジトリを指し示します。
[キーエイリアス (オプション)]	上記のキーストアエイリアスで指定したキーストアに保存されている必要がある、Integration Server の秘密鍵および関連する認証のエイリアス。 Java keytool のようなサードパーティの認証ユーティリティを使用してキーエイリアスを作成する必要があります。Integration Server Administrator ではキーエイリアスは作成できません。

パラメータ	指定内容
[再試行サーバ]	プライマリリモートサーバが使用できない場合にローカル Integration Server の接続先となるリモートサーバのホスト名または IP アドレス (たとえば、workstation6.webmethods.com)。指定する再試行サーバでは、プライマリリモートサーバと同じポートが使用されます。リモートサーバがクラスタの一部である場合、ローカル Integration Server はデフォルトで、再試行サーバへの接続を試行する前に、クラスタ内の他の Integration Server への接続を試行します。クライアントが pub.remote:invoke サービスを使用してリモートサーバでサービスを実行している場合は、サービスと共に *$retryCluster* 入力パラメータを使用してこのデフォルトの動作を変更することができます。このパラメータを false に設定すると、サービスはクラスタ内の他の Integration Server を使用しなくなります。代わりに、サービスはこのページで指定した再試行サーバの使用を試みます。

5. [**変更内容の保存**] をクリックします。

リモートサーバへの接続のテスト

エイリアスを追加したら、リモートサーバへの接続をテストし、エイリアスに対して指定したポート番号およびホスト名 (または IP アドレス) が、現在稼動中の Integration Server を認識していることを確認します。接続をテストするには、以下の手順に従います。

リモートサーバへの接続をテストするには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [リモートサーバ] に移動します。
3. テストするエイリアスの [**テスト**] 列で をクリックします。

   正しく接続されているかどうかを示す状態行が表示されます。状態行は、既存のエイリアスリストの上に表示されます。

エイリアスの編集

エイリアスに関する情報の更新が必要な場合は、エイリアスを編集して変更を行います。エイリアスを編集するには、以下の手順に従います。

リモートサーバに対するエイリアスを編集するには

1. Integration Server Administrator を開きます。

2. [**外部サーバ**] > [リモートサーバ] に移動します。

3. 編集するエイリアスを見つけて、そのエイリアス名をクリックします。

4. エイリアスの情報を更新します。

5. [**変更内容の保存**] をクリックします。

エイリアスの削除

不要となったリモートサーバのエイリアスは、削除することができます。エイリアスを削除するには、以下の手順に従います。

リモートサーバに対するエイリアスを削除するには

1. Integration Server Administrator を開きます。

2. [**外部サーバ**] > [リモートサーバ] に移動します。

3. 削除するエイリアスを検索し、[**削除**] フィールドの×アイコンをクリックします。アクションの確認を求めるダイアログボックスが表示されます。[**OK**] をクリックして、エイリアスの削除を確認します。

送信要求に対するサードパーティのプロキシサーバの指定

Integration Server でリモートサーバに対する要求を実行すると、特定のターゲットサーバに対して HTTP、HTTPS、FTP または SOCKS 要求が発行されます。たとえば、Integration Server でリモート Integration Server のサービスを呼び出したり、Web サービスを呼び出す Web サービスコネクタを実行したりすることがあります。Integration Server がファイアウォールの内側にあり、これらの HTTP、HTTPS、FTP または SOCKS 要求をサードパーティのプロキシサーバ経由でルーティングする必要がある場合、Integration Server Administrator を使用すると、Integration Server によってこれらの要求がルーティングされる先の 1 つ以上のプロキシサーバを識別することができます。

Integration Server でプロキシサーバを使用する場合、プロキシサーバエイリアスを定義する必要があります。プロキシサーバエイリアスは、要求をルーティングする際に経由するプロキシサーバおよびサーバ上のポートを識別します。

送信要求のタイプ (HTTP、HTTPS、FTP および SOCKS) ごとに 1 つ以上のプロキシサーバエイリアスに要求をルーティングするように Integration Server を設定できます。

プロトコルタイプごとにデフォルトのプロキシサーバを指定できます。pub.client:http、pub.client:ftp.login または pub.client.ftp サービスでプロキシサーバエイリアスが指定されていない場合、Integration Server はデフォルトのプロキシサーバエイリアスを使用します。また、Integration Server では、プロキシエイリアスを指定しない、HTTP/S コンシューマ Web サービスエンドポイントエイリアスに関連付けられた Web サービスコネクタを実行するときにも、デフォルトのプロキシサーバエイリアスを使用します。

注:
SOCKS プロキシサーバエイリアスを設定して有効化すると、HTTP、HTTPS および FTP プロキシサーバエイリアスのフォールバックプロキシサーバとして機能します。SOCKS プロキシがデフォルトのプロキシサーバエイリアスとして設定されている場合でも、Integration Server では SOCKS プロキシエイリアスを介した接続を試みる前に、対応する送信要求について設定済みの HTTP、HTTPS および FTP プロキシサーバエイリアスに優先権を与えます。

Integration Server によるプロキシサーバの使用方法

Integration Server がリモートサーバに要求を送信するとき、ターゲットサーバのドメインがプロキシバイパスとしてリストされていない限り、Integration Server ではプロキシサーバを介して要求をルーティングします。ドメインがプロキシバイパスリストにある場合、Integration Server ではターゲットサーバに直接要求を送信します。ドメインがプロキシバイパスリストにない場合、要求を送信するために Integration Server がどのプロキシサーバを使用するかは、以下の条件によって異なります。

- 要求でプロキシサーバエイリアスが指定されているかどうか
- デフォルトのプロキシサーバエイリアスが存在していて有効であるかどうか

次の表は、Integration Server が、要求を送信するときに、使用するプロキシサーバをどのように決定するかを示しています。このプロセスは、HTTP、HTTPS、FTP および SOCKS プロトコルに適用されます。

プロキシサーバエイリアスが指定されているか	デフォルトのプロキシサーバエイリアスが存在するか	Integration Server によるアクション
はい	該当なし	指定されたプロキシサーバエイリアスが有効である場合、Integration Server では、指定されたプロキシエイリアスのプロキシサーバを使用して要求を送信します。送信試行が失敗した場合、Integration Server ではリモートサーバへの直接接続を試行しません。また、Integration Server はデフォルトのプロキシエイリアスまたはその他のすべてのプロキシサーバで指定されたプロキシサーバを使用して要求を送信しません。 指定されたプロキシサーバエイリアスが無効である場合、送信要求は失敗します。
いいえ	はい	デフォルトのプロキシサーバエイリアスが有効である場合、Integration Server では、デフォルトのプロキシサーバエイリアスのプロキシサーバを使用して要求を送信します。送信試行が失敗した場合、Integration Server では、watt.net.proxy.fallbackToDirectConnection パラメータに指定された設定に基づいて、直接接続を使用してリ

プロキシサーバエイリアスが指定されているか	デフォルトのプロキシサーバエイリアスが存在するか	Integration Server によるアクション
		モートサーバに要求を送信するか、例外をスローします。 デフォルトのプロキシサーバエイリアスが有効でない場合、Integration Server では、設定済みプロキシサーバエイリアスのいずれかのプロキシサーバを使用して要求を送信します。そのプロキシサーバを使用して行われた要求の送信試行が失敗した場合、Integration Server は別のプロキシサーバエイリアスを使用して送信を試行します。Integration Server では、要求が送信されるか、すべてのプロキシサーバを試行したが失敗するまで、有効な各プロキシサーバエイリアスを (順序の指定なしで) 使用して試行を継続します。すべてのプロキシサーバが失敗した後、Integration Server では、watt.net.proxy.fallbackToDirectConnection パラメータに指定された設定に基づいて、直接接続を使用してリモートサーバに要求を送信するか、例外をスローします。
いいえ	いいえ	watt.net.proxy.useNonDefaultProxies パラメータに指定された値により異なります。 watt.net.proxy.useNonDefaultProxies パラメータが「true」に設定されている場合、Integration Server では、要求が正常に送信されるか、すべてのプロキシサーバを試行したが失敗するまで、有効な各プロキシサーバエイリアスを (順序の指定なしで) 使用して送信要求を行います。すべてのプロキシサーバが失敗した後、Integration Server では、watt.net.proxy.fallbackToDirectConnection パラメータに指定された設定に基づいて、直接接続を使用してリモートサーバに要求を送信するか、例外をスローします。 指定されたプロトコルのプロキシサーバエイリアスが存在しない場合、Integration Server では、watt.net.proxy.fallbackToDirectConnection パラメータに指定された設定に基づいて、直接接続を使用してリモートサーバに要求を送信するか、例外をスローします。 watt.net.proxy.useNonDefaultProxies パラメータが「false」に設定されている場合、Integration Server では、有効なプロキシサーバエイリアスを使用した送信要求を試行しません。Integration Server は、直接接続を使用してリモートサーバに要求を送信します。

プロキシサーバエイリアスの作成

プロキシサーバエイリアス名は、プロトコル全体で一意である必要があります。つまり、異なるプロトコルに同じ名前のプロキシサーバエイリアスを設定することはできません。たとえば、「myProxy」という名前のプロキシサーバエイリアスは、HTTP、HTTPS、FTP または SOCKS プロトコル全体で 1 つのみ作成できます。

プロキシサーバエイリアスを作成するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [プロキシサーバ] に移動します。
3. [**プロキシサーバエイリアスの作成**] をクリックします。
4. [**プロキシサーバエイリアスのプロパティ**] で、以下のフィールドに情報を入力します。

パラメータ	指定内容
[エイリアス]	このホスト/ポートの組み合わせに使用するエイリアス名。
[ホスト名または IP アドレス]	プロキシサーバのホスト名または IP アドレス。
[ポート番号]	このプロキシサーバが要求を受信待機するポート。
[ユーザ名 (オプション)]	このプロキシサーバにアクセスする際に Integration Server で使用する必要があるユーザ名。
[パスワード (オプション)]	このプロキシサーバにアクセスする際に Integration Server で使用する必要があるパスワード名。
[プロトコル]	ホスト/ポートの組み合わせに使用するプロトコルのタイプ (HTTP、HTTPS、FTP または SOCKS)。
[プロキシタイプ]	[**FTP**] を選択した場合、**プロキシタイプ**の設定に役立つ追加のパラメータが表示されます。要求がプロキシサーバから FTP サーバに転送される場合、以下の情報を指定する必要があります。 ■ FTP サーバの名前 ■ FTP サーバ上のユーザ名 ■ そのユーザのパスワード このような情報を FTP プロキシサーバに送信する方法は、使用するプロキシサーバのタイプによって異なります。Integration Server では、以下のタイプのプロキシサーバをサポートしています。

パラメータ	指定内容

	0.プロキシなし
	FTP プロキシサーバは使用されません。これがデフォルトです。
	1. ftp_user@ftp_host プロキシ権限なし
	プロキシサーバに接続されますが、ログインしません。接続後、以下を送信します。
	`USER ftp_user@ftp_host`
	`PASS ftp_password`
	2. ftp_user@ftp_host プロキシ権限
	プロキシサーバに接続されるので、以下を指定してサーバにログインします。
	`USER proxy_user`
	`PASS proxy_password`
	次に、 以下を送信します。
	`USER ftp_user@ftp_host`
	`PASS ftp_password`
	3. site コマンド
	プロキシサーバに接続されるので、以下を指定してサーバにログインします。
	`USER proxy_user`
	`PASS proxy_password`
	次に、 以下を送信します。
	`SITE ftp_host`
	`USER ftp_user`
	`PASS ftp_password`
	4. open コマンド
	プロキシサーバに接続されるので、以下を指定してサーバにログインします。
	`USER proxy_user`
	`PASS proxy_password`
	次に、 以下を送信します。
	`OPEN ftp_host`
	`USER ftp_user`
	`PASS ftp _password`
	5. ftp_user@proxy_user@ftp_host

パラメータ	指定内容

	プロキシサーバに接続されるので、サーバにログインします。次に、以下を送信します。
	`USER ftp_user@proxy_user@ftp_host`
	`PASS ftp_password@proxy_password`
	6. proxy_user@ftp_host
	プロキシサーバに接続されるので、以下を指定してサーバにログインします。
	`USER proxy_user@ftp_host`
	`PASS proxy_password`
	次に、 以下を送信します。
	`USER ftp_user`
	`PASS ftp_password`
	7. ftp_user@ftp_host proxy_user
	プロキシサーバに接続されますが、ログインしません。次に、以下を送信します。
	`USER ftp_user@ftp_host proxy_user`
	`PASS ftp_password`
	`ACCT proxy_password`

[SOCKS バージョン]	プロトコルとして [**SOCKS**] を選択した場合、SOCKS バージョンの設定に役立つ追加のパラメータが表示されます。プロキシサーバに接続するときに使用する SOCKS プロトコルのバージョンに基づいて、[**SOCKS v4**] または [**SOCKS v5**] を選択します。 **注:** SOCKS プロトコルバージョン 4 では認証をサポートしていません。SOCKS プロトコルバージョン 4 を選択する場合は、認証クレデンシャル (ユーザ名およびパスワード) を指定しないでください。
デフォルト	このプロキシサーバエイリアスをそのプロトコルタイプのデフォルトのプロキシサーバエイリアスにするかしないか。[はい] または [いいえ] をクリックします。デフォルトのプロキシサーバエイリアスはプロトコルタイプごとに 1 つのみ設定できます。

5. [**変更内容の保存**] をクリックします。

プロキシサーバエイリアスの編集

特定のプロキシサーバエイリアスのプロパティ変更が必要な場合は、そのプロキシサーバエイリアスを編集することができます。

プロキシサーバエイリアスを編集するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [プロキシサーバ] に移動します。
3. [**プロキシサーバリスト**] テーブルの [**エイリアス**] 列で、編集するプロキシサーバエイリアスの名前をクリックします。

   [**<プロキシサーバエイリアス> の編集**] ページが表示されます。
4. 必要な [**プロキシサーバエイリアスのプロパティ**] を変更します。
5. [**変更内容の保存**] をクリックします。

プロキシサーバエイリアスの無効化

プロキシサーバエイリアスを無効にするには、以下の手順に従います。

プロキシサーバエイリアスを無効にするには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [プロキシサーバ] に移動します。
3. [プロキシサーバ] のメインページで、[**プロキシサーバリスト**] の [**有効**] 列にある [**はい**] をクリックします。

   Integration Server によって、アクションの確認を求めるダイアログボックスが表示されます。
4. [**OK**] をクリックして、プロキシサーバエイリアスを無効にします。

プロキシサーバエイリアスの有効化

プロキシサーバエイリアスを有効にするには、以下の手順に従います。

プロキシサーバエイリアスを有効にするには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [プロキシサーバ] に移動します。
3. [プロキシサーバ] のメインページで、[**プロキシサーバリスト**] の [**有効**] 列にある [**いいえ**] をクリックします。

Integration Server によって、アクションの確認を求めるダイアログボックスが表示されます。

4. [**OK**] をクリックして、プロキシサーバエイリアスを有効にします。

デフォルトのプロキシサーバエイリアスの指定

プロトコルタイプごとにデフォルトのプロキシサーバエイリアスを指定できます。pub.client:http、pub.client:ftp.login または pub.client.ftp サービスでプロキシサーバエイリアスが指定されていない場合、Integration Server ではデフォルトのプロキシサーバエイリアスを使用します。また、Integration Server では、HTTP または HTTPS プロトコルのコンシューマ Web サービスエンドポイントエイリアスでプロキシサーバエイリアスが指定されていない場合も、デフォルトのプロキシサーバエイリアスを使用します。

任意のプロキシサーバエイリアスをデフォルトエイリアスにすることができます。ただし、特定のプロトコルタイプに対してデフォルトのプロキシは 1 つしか設定できません。

プロトコルに対してデフォルトのプロキシサーバエイリアスを選択するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [プロキシサーバ] に移動します。
3. [**プロキシサーバリスト**] テーブルの [**エイリアス**] 列で、デフォルトとして使用するプロキシサーバエイリアスの名前をクリックします。
4. [**プロキシサーバエイリアスのプロパティ**] で、[**デフォルト**] セクションにある [**はい**] をクリックして、このプロキシサーバを新しいデフォルトのプロキシサーバにします。
5. [**変更内容の保存**] をクリックします。

   別のプロキシサーバエイリアスが既にそのプロトコルのデフォルトのプロキシサーバエイリアスとして設定されている場合、Integration Server によって変更の確認を求めるダイアログボックスが表示されます。
6. [**OK**] をクリックして、デフォルトのプロキシサーバエイリアスを変更します。

プロキシサーバエイリアスの削除

プロキシサーバエイリアスが不要になった場合は、削除できます。

プロキシエイリアスを削除するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [プロキシサーバ] に移動します。

3. [**プロキシサーバリスト**] で削除するエイリアスを検索し、[**削除**] 列の ✕ アイコンをクリックします。

   Integration Server によって、アクションの確認を求めるダイアログボックスが表示されます。

4. [**OK**] をクリックして、選択したプロキシサーバエイリアスを削除します。

プロキシサーバのバイパス

送信する HTTP、HTTPS、FTP または SOCKS 要求に対してプロキシサーバを使用する場合、オプションとして、プロキシをバイパスして、選択した要求を直接そのターゲットにルーティングできます。

バイパスするには、Integration Server Administrator を使用して、Integration Server で要求を直接発行する宛先のドメインリストを定義します。

プロキシバイパスアドレスを指定するには

1. Integration Server Administrator を開きます。

2. [**外部サーバ**] > [プロキシサーバ] に移動します。

3. [**プロキシのバイパスの編集**] をクリックします。

4. [**アドレス**] フィールドに、Integration Server で要求を直接発行する宛先の各サーバのドメインとホストを完全修飾名で入力します。ホスト名およびドメイン名は、サーバで使用する URL の表示どおりに正確に入力してください。複数の名前を入力するには、それぞれの名前をカンマで区切ります。

   アスタリスク (*) を使用することにより、似た名前を持つサーバを複数指定できます。アスタリスクは複数の文字に相当します。たとえば、localhost、www.yahoo.com、home.microsoft.com およびホスト名が NYC で始まるすべてのホストに対する要求をバイパスする場合は、次のように入力します。

   ```
   localhost,www.yahoo.com,home.microsoft.com, NYC*.*
   ```

5. [**変更内容の保存**] をクリックします。

   プロキシバイパスアドレスが、[**プロキシサーバ**] のメインページの [プロキシのバイパス] に表示されます。

Integration Server によるログ、状態、その他の情報の保存場所の設定

Integration Server は、内部処理 (スケジュール済みジョブ、保証付きデリバー、トリガーの結合)、監査とロギング、セントラルユーザ管理、ドキュメント履歴、プロセス保全など、さまざ

まな Integration Server の領域についてデータを収集および保存します。収集されたデータは、機能エイリアスを介して Integration Server に識別されるさまざまなデータベースコンポーネントに保存されます。それぞれのデータベースコンポーネントに保存されるデータの詳細および機能エイリアスの設定手順については、157 ページの「Integration Server を外部 RDBMS のデータベースコンポーネントへ接続する」を参照してください。

組み込みデータベースから外部 RDBMS への切り替え

組み込みデータベースを指定して Integration Server をインストールした場合に、後で外部 RDBMS に切り替えることができます。

組み込みデータベースから外部 RDBMS に切り替えるには

1. Integration Server Administrator を開きます。
2. [**セキュリティ**] > [認証] > [クライアント認証の設定] をクリックし、認証マッピングを書き留めます。
3. IS Internal および Cross Reference データベースコンポーネントを作成し、JDBC 接続プールに接続します。手順については、157 ページの「Integration Server を外部 RDBMS のデータベースコンポーネントへ接続する」を参照してください。

   注:
   データベースコンポーネントを JDBC 接続プールに接続すると、Integration Server によって外部 RDBMS への書き込みが開始されます。プロパティを設定する必要はありません。
4. マイグレーションユーティリティ pub.scheduler:migrateTasksToJDBC を実行して、スケジュール済みタスクを組み込みデータベースから外部 RDBMS に移行します。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

   注:
   このサービスは、スケジュール済みタスクのみを移行します。組み込みデータベースに保存されている認証マッピングおよびランタイムデータは移行されません。
5. Integration Server Administrator で、[**セキュリティ**] > [認証] > [クライアント認証の設定] ページに移動し、認証マッピングを再指定します。手順については、553 ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。

拡張設定の使い方

特殊なサーバの設定を表示したいこともあります。サーバ設定パラメータは server.cnf ファイルに保存されますが、Integration Server Administrator の [**設定**] > [拡張設定] ページで表示し編集できます。Integration Server Administrator には、パラメータごとに現在の値およびデフォルト値が表示されます。また、Integration Server Administrator では、パラメータの変更を適用するには再起動が必要かどうかも示されます。

[**サーバ**] > [拡張設定] ページで、次の 1 つ以上の処理を行います。

- [**検索**] を使用して、拡張設定 (つまり、サーバ設定パラメータ) の表示リストをフィルタします。[**検索**] をクリックして [**拡張設定の検索**] を開き、次の作業を行います。
  - 設定名の一部または全体を使用して 1 つ以上の設定を検索します。たとえば、名前に「jms」を含むすべての設定をリストするには、[**キー名**] に「jms」と入力し、[**適用**] をクリックします。
  - デフォルト値を使用しない設定を検索するには、[**システムデフォルトから変更した設定を検索する**] を選択します。
  - お気に入りの設定のみを表示するには、[**スターなし設定を非表示にする**] チェックボックスをオンにします。
- 拡張設定を編集するには、設定行で ⋮ > [**編集**]. をクリックします。[**拡張設定の編集**] ダイアログボックスで新しい値を指定し、[**保存**] をクリックします。変更を適用するにはサーバの再起動が必要な場合、Integration Server Administrator に通知が表示されます。
- 拡張設定を追加するには、⊕ をクリックします。[**拡張設定の追加**] ダイアログボックスで、設定の名前および使用する値を指定します。[**保存**] をクリックして、設定と値を server.cnf ファイルに保存します。Integration Server に修正を適用する操作の一部として、拡張設定の追加が必要なことがあります。
- 一度に複数の拡張設定を追加するには、テーブルの上部にある ⋮ > をクリックし、[**バッチ追加**] を選択します。[**複数の拡張設定の追加**] ダイアログボックスで、key=value 形式で各設定をそれ自体の行に入力します。[**保存**] をクリックして、設定と値を server.cnf ファイルに保存します。
- 設定をデフォルト値にリセットするには、設定行で ⋮ > [**デフォルトにリセット**] をクリックします。
- 拡張設定を削除するには、設定行で ⋮ > [**削除**] をクリックします。削除したシステム設定は、Integration Server の再起動時にデフォルト値で復元されます。
- 一度に複数の設定で同じアクションを実行するには、設定名の横のチェックボックスをオンにします。次に、⋮ をクリックし、適用するアクションを選択します。

重要:
通常は、Integration Server Administrator を使用して server.cnf ファイルのプロパティを設定しますが、テキストエディタを使用してファイルを直接編集する必要がある場合もあります。このファイルを直接更新する前に、必ず Integration Server をシャットダウンしてください。

文字エンコーディングの指定

他のアプリケーションとの相互運用性を確保するために、Integration Server では、文字エンコーディングについて複数の形式をサポートしています。次の表は、デフォルトの設定およびその設定を管理するサーバのプロパティを示したものです。

[アクション]	デフォルト設定	管理するプロパティ
テキストファイルの読み取りと書き込み	JVM の file.encoding プロパティ	watt.server.fileEncoding
ネットワークからのテキストの読み取りとネットワークへのテキストの書き込み	UTF-8	watt.server.netEncoding

重要:
これらの設定を変更する場合は、事前に Software AG Global Supportまでご連絡ください。ほとんどの場合、デフォルトの設定で使用することができます。設定を間違えると、予測不可能な結果を招く恐れがあります。

JVM の設定

Integration Server を実行する JVM の設定は、custom_wrapper.conf に含まれるプロパティで指定します。これらの設定は、JVM を起動するときに Java サービスラッパーに渡されます。JVM 設定を更新する手順については、以下のトピックで説明します。

- 148 ページの「Integration Server の JDK または JRE の指定」
- 149 ページの「Integration Server が使用する JVM ヒープサイズの変更」
- 52 ページの「Integration Server への Java システムプロパティの受け渡し 」

Java サービスラッパーの詳細については、『*Software AG Infrastructure Administrator's Guide*』を参照してください。

Integration Server の JDK または JRE の指定

Integration Server は、JDK または JRE を指している必要があります。デフォルトでは、Integration Server は Integration Server のインストールと同時にインストールされる JDK の場所を指しています。必要に応じて、別の場所を指定できます。

Integration Server の Java の場所を指定する前に、JDK または JRE の場所を指定する必要があるかどうかを判断します。Designer を使用して Integration Server 上の Java サービスを開発およびコンパイルする場合は、JDK の場所を指定します。Java サービスのコンパイルに Integration Server のこのインストールを使用しない場合は、JRE の場所を指定できます。

重要:
別の JDK または JRE を指定する場合、Software AG Installer が Integration Server と共にインストールした JDK または JRE を削除しないでください。Integration Server と共にインストールされた JDK および JRE は、Software AG Uninstaller を実行するときに必要になります。

注:
Microservices Runtime は Java Service Wrapper、wrapper.conf ファイル、または custom_wrapper.conf を使用しません。Microservices Runtime 用の Java の場所を変更するに

は、テキストエディタを使用して *Integration Server_directory* /bin/setenv.bat(sh) を開き、JAVA_DIR プロパティを JDK または JRE の場所に設定します。

Integration Server の Java の場所を指定するには

1. custom_wrapper.conf ファイルをテキストエディタで開きます。custom_wrapper.conf ファイルは次の場所にあります。

   Software AG_directory \profiles\IS_*instance_name*\configuration

2. `wrapper.java.command` プロパティを追加して、JDK または JRE インストールディレクトリの場所を指定するように設定します。次に例を示します。

   ```
   wrapper.java.command=C:\SoftwareAG\jvm\jvm\bin\java
   ```

   注:
   また、UNIX システムで Integration Server を実行している場合は、ライブラリパスプロパティが JDK または JRE インストールディレクトリの場所を示すように変更する必要があります。ライブラリパスは、AIX では `LIBPATH`、その他のすべての Linux の種類では `LD_LIBRARY_PATH` で表されます。

3. ファイルを保存して閉じます。

4. Integration Server を再起動します。

注:
Java の場所を変更し、Integration Server を使用して Java サービスを開発およびコンパイルする場合は、watt.server.compile 設定パラメータの値も変更する必要があります。watt.server.compile の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

Integration Server が使用する JVM ヒープサイズの変更

JVM のヒープまたはオンヒープのサイズは、サーバプロセスへのメモリの割り当て量を示します。いずれかの時点で、Integration Server が使用する JVM がメモリ不足にならないように、ヒープサイズの最小値と最大値の増量が必要となる場合があります。サーバがドキュメントをパブリッシュおよびサブスクライブするように設定する場合、およびオンヒープキャッシュを設定する場合には、ヒープサイズを検討する必要があります。

ヒープサイズは、custom_wrapper.conf ファイル内で指定される次の Java プロパティにより制御されます。

プロパティ	説明
`wrapper.java.initmemory`	最小ヒープサイズ。デフォルト値は 256 MB です。
`wrapper.java.maxmemory`	最大ヒープサイズ。デフォルト値は 1024 MB です。

ヒープサイズの最大値と最小値を増やす必要があるかどうかは、容量計画とパフォーマンス分析に基づいて判断してください。

注:
Microservices Runtime では Java Service Wrapper および custom_wrapper.conf ファイルを使用しません。Microservices Runtime のメモリ設定を変更するには、テキストエディタを使用して *Integration Server_directory* /bin/setenv.bat(sh) で JAVA_MIN_MEM および JAVA_MAX_MEM プロパティを設定します。

ヒープサイズを変更するには

1. custom_wrapper.conf ファイルをテキストエディタで開きます。custom_wrapper.conf ファイルは次の場所にあります。

   Software AG_directory \profiles\IS_*instance_name*\configuration

2. `wrapper.java.initmemory` および `wrapper.java.maxmemory` パラメータを設定して、Integration Server で必要とされる最小および最大ヒープサイズを指定します。次に例を示します。

   ```
   wrapper.java.initmemory=256
   wrapper.java.maxmemory=512
   ```

3. ファイルを保存して閉じます。

4. Integration Server を再起動します。

Integration Server のアセット情報のパブリッシュおよび取り消し

情報またはメタデータを CentraSite に置かれている共有ライブラリまたはレジストリにパブリッシュできます。このメタデータをパブリッシュすることによって、これらのアセットを他の CentraSite ユーザからアクセスできるようにします。パブリッシュしたメタデータを CentraSite から取り消すこともできます。

パブリッシュできるメタデータは、Integration Server アセット、Integration Server 管理アセット、Trading Networks (TN) ドキュメントタイプに関するものです。Integration Server 管理アセットには、アダプタ接続、Broker 接続、JMS 接続エイリアスが含まれています。パブリッシュできるアセットの一覧については、『*webMethods Service Development Help*』を参照してください。

Software AG Designer を使用して、CentraSite との間でメタデータのパブリッシュや取り消しを行うアセットを選択できます。Designer では、メタデータは、Designer 内の CentraSite 接続で定義されたクレデンシャルを使用してパブリッシュされます。

WmAssetPublisher パッケージの pub.metadata.assets:publishPackages サービスを使用して、パッケージおよび管理アセットのメタデータをパブリッシュすることもできます。このサービスを使用するスケジュール済みタスクを作成して、メタデータを定期的にパブリッシュできます。pub.metadata.assets:publishPackages サービスでは、メタデータは Integration Server で定義された CentraSite クレデンシャルを使用してパブリッシュされます。

注:
Integration Server のアセットに関するメタデータをパブリッシュする前に、CentraSite に接続するように Integration Server を設定する必要があります。手順については、151 ページの「CentraSite に接続するための Integration Server の設定」を参照してください。

メタデータのパブリッシュおよび取り消しを行う方法の詳細については、『*webMethods Service Development Help*』を参照してください。CentraSite 内の共有ライブラリの管理については、CentraSite のマニュアルを参照してください。

CentraSite に接続するための Integration Server の設定

pub.metadata.assets:publishPackages サービスを使用してメタデータをパブリッシュする前に、CentraSite の共有ライブラリに接続するように Integration Server を設定する必要があります。

CentraSite に接続するために Integration Server を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動し、**WmAssetPublisher** に対応する アイコンをクリックします。
3. [**設定の編集**] をクリックします。
4. [**メタデータライブラリの設定**] で、以下のフィールドに情報を入力します。

パラメータ	指定する値
[IS 識別子]	Integration Server が稼動しているマシンの名前または IP アドレス。デフォルトでは、Integration Server によってこのフィールドにマシンの IP アドレスが移入されます。ただし、ここで Integration Server がエイリアスとして処理する名前を入力することができます。さらに、この名前は Integration Server によってパブリッシュされるメタデータに含められます。任意の名前を使用できますが、次の文字は使用できません。 #-&@^!%*:$./\\`;,~+=)(|}{[][><"
[CentraSite URL]	Integration Server のアセットに関するメタデータをパブリッシュする宛先の CentraSite の URL。
ユーザ名	メタデータのパブリッシュおよび撤回に使用する CentraSite 上のユーザアカウントの名前。
[パスワード]	**ユーザ名**に対応するパスワード。

5. [**変更内容の保存**] をクリックします。

設定した接続をテストし、指定した詳細を使用して Integration Server から CentraSite に正常に接続できることを確認できます。手順については、151 ページの「CentraSite に接続するための Integration Server の設定」を参照してください。

CentraSite への接続のテスト

CentraSite に接続するように Integration Server を設定した後、指定したホスト名、IP アドレスおよびライブラリ名で現在稼動している CentraSite ライブラリが識別されることを確認するために、接続をテストできます。

CentraSite への接続をテストするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動し、**WmAssetPublisher** に対応する アイコンをクリックします。
3. [**接続テスト**] をクリックします。

   Integration Server Administrator によって、正しく接続されているかどうかを示す状態行が表示されます。状態行は、ページの一番上に表示されます。

リモートクライアント JMX 監視のポート設定

Integration Server を使用すると、リモートクライアントから JMX 監視を使用できます。JMX 監視はデフォルトで有効ですが、監視する Integration Server の com.software.jmx.connector.pid-*port*.properties ファイルで JMX 監視リモートポートを設定することも可能です。「*port*」は現在の JMX 監視ポートです。JMX 監視を無効化することはできません。

注:
デフォルトで、Microservices Runtime 用の JMX ポートは有効になっていません。Microservices Runtime 用の JMX ポートを有効にするには、 *Integration Server_directory* /bin/setenv.bat(sh) を変更して `JMX_ENABLED=true` および `JMX_PORT=9192` を含めます。詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

リモートクライアント JMX 監視のポートを設定するには

1. リモート監視を行う Integration Server 上で、com.software.jmx.connector.pid-*port*.properties ファイルをテキストエディタで開きます。com.software.jmx.connector.pid-*port*.properties ファイルは次の場所にあります。

   Software AG_directory
   \profiles\IS_*instance_name*\configuration\com.softwareag.platform.config.propsloader

2. `port` プロパティで JMX ポートのポート番号を指定します。これは、デフォルトでは「`8075`」です。

3. Integration Server を再起動します。

起動中のデバッグ接続を受け入れるためのIntegration Server設定

Integration Server を使用すると、デバッグツールを接続し、Integration Server 上で実行されている Java サービスをリモートでデバッグできます。デバッガのアタッチ用に Integration Server が受信待機するポートを指定できます。

Software AG Designer を使用して Integration Server 上で実行されている Java サービスをデバッグする方法の詳細については、『*Software AG Designer Online Help*』を参照してください。

注:
デフォルトで、Microservices Runtime 用の JMX ポートは有効になっていません。Microservices Runtime 用の JMX ポートを有効にするには、 *Integration Server_directory* /bin/setenv.bat(sh) を変更して `JMX_ENABLED=true` および `JMX_PORT=9192` を含めます。

起動中にデバッグ接続を受け入れるように Integration Server を設定するには

1. > [**シャットダウンまたは再起動**] をクリックします。

2. [**シャットダウン**] をクリックします。

3. ポートの変更が必要な場合は、以下を実行します。

   a. startDebugMode.bat ファイルをテキストエディタで開きます。startDebugMode.bat ファイルは次の場所にあります。

      Software AG_directory \profiles\IS_*instance_name*\bin

      注:
      Microservices Runtime の場合、 *Integration Server_directory* /bin/startDebugMode.bat(sh) を開きます。

   b. デバッガのアタッチ用にサーバが受信待機するポートを指定するために、DEBUG_PORT プロパティを見つけて変更します。デフォルトは 10033 です。

   c. デバッガのアタッチまで JVM を待機させる場合、SUSPEND_MODE プロパティを見つけて y に設定します。

   d. 変更を保存して startDebugMode.bat ファイルを閉じます。

4. Integration Server とデバッグツールが異なるマシン上にあり、ファイアウォールポートが必要な場合は、デバッグポートのファイアウォールポートを開きます。

5. startDebugMode.bat を実行します。

Integration Server によってコンソールに以下が記録されます。

```
"Debug enabled (portNumber)"
Listening for transport dt_socket at address: portNumber
```

Integration Server が再起動します。

Integration Server での CORS の使用

CORS (クロスオリジンリソース共有) は、あるオリジンで実行されているクライアント側の Web アプリケーションが別のオリジンのリソースにアクセスできるようにするために、ユーザエージェント (Web ブラウザなど) の規格を提供する仕様です。セキュリティ上の理由から、スクリプトによって開始されたクロスオリジン要求は、通常、ユーザエージェントによって制限されます。ただし、ユーザエージェントは CORS を使用して、これらのクロスオリジン通信を許可および定義できます。

Integration Server による CORS 要求の処理方法

Integration Server には、CORS 要求の処理を有効化および設定するための拡張設定のセットがあります。CORS 要求には、要求を識別して Integration Server と通信するために使用できる HTTP ヘッダーのセットが含まれています。CORS のサポートが有効な場合、Integration Server はすべての受信要求についてこれらのヘッダーをチェックします。要求に CORS ヘッダーが含まれている場合、Integration Server では要求の妥当性検査を試行します。Integration Server は有効な要求を処理し、CORS 応答ヘッダーを応答に含めます。

注:
CORS を使用するには、CORS 規格で定義されているように、サーバとクライアントの両方が CORS ヘッダーによって通信する必要があります。

Integration Server の CORS 要求の受け入れの設定

CORS 要求を受け入れて処理するように Integration Server を設定するには、以下の手順に従います。

Integration Server での CORS 処理を有効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [拡張設定] に移動します。
3. アイコンをクリックします。
4. [拡張設定の追加] ウィンドウで、キー名を「watt.server.cors.enabled」、値を「`true`」と入力し、[**保存**] をクリックします。

5. Integration Server がリソースにアクセスするためクロスオリジン要求を許可する必要がある URI を指定するには、ステップ 3 と 4 を繰り返し、キーとして watt.server.cors.allowedOrigin を追加し、次のいずれかの方法で値を指定します。

   - `<protocol>://<hostname>` または `<protocol>://<hostname>:<port>`この *<protocol>* は HTTP または HTTPS、*<hostname>* はマシンの IP アドレスまたは名前、*<port>* はポート番号です。
   - 許可された URI を含むテキストファイル。

     例: `file:c:\cors\allowedOriginsList.txt`

     注:
     ファイルでは、watt.sever.cors.allowedOrigins の値として指定された URI に必要な構文と同じ構文を使用する必要があります。ただし、各行を許可された元のサーバ URI または正規表現にする必要がある点が異なります。

   値の大文字と小文字は区別されます。複数の値を指定する場合は、スペースまたはカンマ区切り文字を使用します。アスタリスク (*) を使用して、任意の URI またはオリジンを使用できるように指定できます。また、許可されたオリジンサーバのカンマ区切りリストで正規表現を使用することもできます。watt.server.cors.allowedOrigins サービス設定パラメータの設定の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

6. オプションで、CORS 処理の以下のパラメータを設定します。

目的	入力内容
CORS 要求への応答で、Integration Server が CORS Access-Control-Expose-Headers ヘッダーに含めることができる値を指定します。	`watt.server.cors.exposedHeaders=<value1,value2,value3>` *value#* はアプリケーションがアクセスできる応答ヘッダーです。クライアントサイドコードを確認し、応答ヘッダーがある場合はクライアントによって取得され、公開する必要がある応答ヘッダーを判別します。
クライアントが Integration Server にクロスオリジン要求を送信できるホストおよびポートを指定します。	`watt.server.cors.host=<hostname>:<port>`
ユーザエージェントがプリフライト要求の結果をキャッシュできる時間 (秒) を指定します。	`watt.server.cors.maxAge= <number of seconds>`
すべての CORS 要求への応答の CORS Access-Control-Allow-Credentials ヘッダーを Integration Server によって設定します。	`watt.server.cors.supportsCredentials=true`

目的	入力内容
Integration Server がクロスオリジン要求で許可する要求ヘッダーを指定します。Integration Server は、プリフライト要求への応答時にこれらのヘッダーを Access-Control-Allow-Headers 応答ヘッダーの値として含めます。	`watt.server.cors.supportedHeaders=<header1,header2,header3>` *header#* は、クライアントが実際の要求に含めることができるヘッダーです。サーバサイドコードを確認し、応答ヘッダーがある場合は、どの応答ヘッダーがサーバアプリケーションによって読み込まれ、明示的に許可される必要があるのかを判断します。
Integration Server がクロスオリジン要求で許可する HTTP メソッドを指定します。	`watt.server.cors.supportedMethods=<method1,method2,method3>` 指定可能な値は、OPTIONS、HEAD、GET、POST、PUT、PATCH および DELETE です。Integration Server は、デフォルトでこれらのいずれの値も受け入れます。

これらのパラメータの重要な使用方法および詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

7. [**変更内容の保存**] をクリックします。

8. watt.server.cors.allowedOrigins の値としてテキストファイルを使用した場合、URL http://*host*:*port*/invoke/wm.server.admin:refreshAllowedOrigins を呼び出し、参照先のテキストファイルを Integration Server にロードする内部サービスを実行します。

   ここで *host* および *port* は Integration Server のホストおよびポートです。

9 Integration Server を外部 RDBMS のデータベースコンポーネントへ接続する

- 概念 158
- 接続プールに機能をポイントする 158
- MySQL Community Edition データベースドライバのインストール 159
- Tibero JDBC ドライバのインストール 160
- ドライバエイリアス定義の作成 160
- 接続プールの作成 160
- データベースへの送信接続にケルベロス認証を使用する 165

概念

Integration Server を次のように外部 RDBMS のデータベースコンポーネントに接続します。

- JDBC データベース接続プールを定義します。接続プールは、データベースコンポーネントをホストするデータベースサーバに Integration Server を接続するためのパラメータを指定します。Integration Server は、実行時に、各データベースコンポーネントの接続プールのインスタンスを個別に作成します。
- データベースコンポーネントの接続プールで各機能をポイントすることによって、データベースコンポーネントに書き込むように機能に指示します。たとえば、ISCoreAudit Log 機能を ISCoreAudit データベースコンポーネントの接続プールにポイントし、Xref 機能を CrossReference データベースコンポーネントの接続プールにポイントするなどです。Integration Server では、データベースコンポーネントに書き込み可能なデータのタイプごとに事前定義済みの機能が用意されています。

Integration Server のインストール中に外部データベースの使用を選択すると、Software AG Installer はデータベース接続パラメータを指定するよう求めます。初めて Integration Server を起動したときに、Integration Server は指定されたパラメータから JDBC 接続プールを作成し、その接続プールを使用するデータベースコンポーネントに書き込むように事前定義済みの機能を設定します。

次の表に、Integration Server に含まれている事前定義済みの機能とそのデータベースコンポーネントを示します。

機能	データベースコンポーネント
ISInternal	ISInternal および DistributedLocking
ISCoreAudit	ISCoreAuditLog
Xref	CrossReference
DocumentHistory	DocumentHistory
ProcessAudit	ProcessAudit
Process Engine	ProcessEngine

機能を追加または削除することはできませんが、接続プールを作成して機能を新しいプールにポイントすることができます。

接続プールに機能をポイントする

次の手順を参照して、機能を接続プールにポイントできます。

接続プールに機能をポイントするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [JDBC プール] に移動します。

3. [機能エイリアスの定義] 領域で、その機能の [編集] をクリックします。

4. [関連付けられたプールエイリアス] フィールドで、機能をポイントする接続プールを識別します。

5. [**フェイルファストモードが有効**] オプションが使用可能な場合は、有効にするか無効にするかを選びます。

   データベースが使用できないために発生した一時的なエラーによって、接続プールがそのデータベースに接続できなくなる可能性があります。この状況を処理するために、フェイルファストモードを入力する機能を設定できます。フェイルファストモードでは、機能によってデータベースを接続しようとすると、直ちに SQLException が返されます。再試行ができないため、時間を節約することができます。データベースへの接続が復元されると、機能はフェイルファストモードを終了し、通常の動作に戻ります。フェイルファストモードは、同期監査ログを使用するとパフォーマンスが向上します (を参照)。

   注:
   フェイルファストは CentralUsers 機能エイリアス定義で使用できません。したがって、CentralUsers 機能エイリアスの編集時に [**フェイルファストモードが有効**] と [**現在フェイルファストモードである**] オプションは表示されません。

6. [**設定の保存**] をクリックします。

7. [機能エイリアスの定義] 領域で、機能の [再起動] をクリックして、プールを初期化します。

8. 機能の ▶ をクリックして、Integration Server がデータベースに接続できることを確認してください。

9. Integration Server を再起動します。

MySQL Community Edition データベースドライバのインストール

MySQL Community Edition データベースドライバをインストールするには、以下の手順を参照してください。

MySQL Community Edition データベースドライバをインストールするには

1. MySQL Community Edition データベースドライバをダウンロードします。

2. ダウンロードファイルを展開し、ドライバ JAR を *Software AG_directory* `\IntegrationServer\lib\jars\custom` にコピーします。

Tibero JDBC ドライバのインストール

Integration Server は、Tibero JDBC ドライバを使用した Tibero データベースインスタンスへの接続をサポートします。

Tibero JDBC ドライバをインストールするには

- Tibero JDBC ドライバ jar ファイルをダウンロードし、以下のディレクトリに配置します。

  Software AG_directory \IntegrationServer\lib\jars\custom

ドライバエイリアス定義の作成

ドライバエイリアス定義は Integration Server にデータベースドライバを示します。Integration Server には既にいくつかのドライバエイリアスが定義されていますが、ドライバエイリアス定義の作成が必要なことがあります。

ドライバエイリアス定義を作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [JDBC プール] > [ドライバエイリアスの作成] に移動します。
3. ドライバエイリアスに関する以下の情報を指定します。

パラメータ	説明
[エイリアス名]	ドライバに使用するエイリアス名。名前は任意の長さに設定でき、任意の文字を使用できます。
[エイリアスの説明]	ドライバエイリアスの簡単な説明。
[ドライバクラス名]	JDBC ドライバの Java クラスの名前。 Tibero JDBC ドライバの場合、クラス名を `com.tmax.tibero.jdbc.TbDriver` に設定します。

4. [**設定の保存**] をクリックします。

接続プールの作成

接続プールは、[プールエイリアス定義のコピー] を使用して既存のプールをコピーするか、このセクションで説明されている新しいフィールドの値を指定することで作成できます。

新しいフィールドの値を指定して接続プールを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [JDBC プール] に移動します。
3. [**プールエイリアス定義の作成**] をクリックし、フィールドを指定します。

パラメータ	指定内容
[エイリアス名]	接続プールに使用する名前。使用しているオペレーティングシステムでファイル名として有効なすべての文字を使用できます。
[エイリアスの説明]	プールの簡単な説明。
[関連付けられたドライバエイリアス]	プールに接続するために使用する Integration Server のデータベースドライバ。

[データベース URL]	データベースサーバの URL。
	■ Trading Networks の場合を除き、すべてのデータベース URL で DataDirect Connect 接続オプション MaxPooledStatements=35 を使用します。この接続オプションにより、作成されたステートメントがキャッシュされるため、パフォーマンスが向上します (Trading Networks では独自のプーリングメカニズムを使用して、作成されたステートメントがキャッシュされます)。
	■ DB2 の場合、Integration Server が指定されたデータベースユーザのデフォルトスキーマ以外のスキーマに接続するときには、URL で次の接続オプションを指定する必要があります。
	`;AlternateId=schema;"InitializationString=(SET CURRENT PATH=current_path,schema)"; MaxPooledStatements=35`
	AlternateID は、動的に作成された SQL ステートメントで未修飾のデータベースオブジェクトの修飾に使用されるデフォルトスキーマの名前です。
	■ MySQL Community Edition の場合、URL に `relaxAutoCommit`、`useLegacyDatetimeCode`、`serverTimezone` の各パラメータの接続オプションを指定する必要があります。たとえば、次のような接続オプションを指定できます。
	`jdbc:mysql://<server>:<3306|port> /databaseName?relaxAutoCommit=true& useLegacyDatetimeCode=false& serverTimezone`
	注:
	URL の `relaxAutoCommit` および `useLegacyDatetimeCode` パラメータは廃止されたため、MySQL Community Edition 8.x ではサポートしていません。

パラメータ	指定内容
	■ Tibero の場合、次のように指定します。 `jdbc:tibero:thin:@<server>:<8629|port>:databaseName`
[診断]	外部データベースとやりとりするために Integration Server によって使用される DataDirect Connect JDBC ドライバのための DataDirect 診断機能。

オプション	説明
[Spy]	DataDirect Connect JDBC ドライバについて、DataDirect Spy 診断機能を有効にします。DataDirect Spy は、Integration Server と外部 RDBMS との間の JDBC 呼び出しおよび SQL ステートメントをログに記録します。
[Snoop]	DataDirect Connect JDBC ドライバについて、DataDirect Snoop ツールを有効にします。DataDirect Snoop ツールは、Integration Server と外部 RDBMS の間のネットワークパケットをログに記録します。作成されたログファイルは、トレースおよび診断の目的で使用することができます。

[Spy 属性]

Integration Server が DataDirect Spy 診断機能で収集された診断データを記録するログファイルの名前および場所。この値によって、DataDirect Spy 属性も定義されます。デフォルト値は以下のとおりです。

```
SpyAttributes=(log=(file)/logs/spy/<alias_name>.log;
logTName=yes;timestamp=yes)
```

<alias_name> は JDBC 接続プールエイリアスの名前です。

注:
通常、デフォルト値を変更する必要はありません。ただし、属性がシステムのニーズを満たさない場合は、値を変更することができます。この診断ツールは *Integration Server_directory* /instances/*instanceName*/logs/spy ディレクトリからデータを収集することに注意してください。ログファイルの場所を変更すると、DataDirect Spy によって収集されるデータを診断ユーティリティがインポートできないことがあります。診断ユーティリティの使用方法の詳細については、995 ページの「診断ユーティリティの使用方法」を参照してください。DataDirect Spy の属性を設定する方法の詳細については、DataDirect Web サイトでマニュアルを参照してください。

注:
MySQL Community Edition はこのオプションをサポートしていません。

パラメータ	指定内容

[Snoop ログ出力パラメータ]	Integration Server が DataDirect Snoop ツールで収集された診断データを記録するログファイルの名前および場所。この値によって、DataDirect Snoop ツール属性も定義されます。デフォルト値は以下のとおりです。
	
	`ddtdbg.ProtocolTraceEnable=true;ddtdbg.Protocol TraceMaxline=16;ddtdbg.ProtocolTraceLocation=/logs/snoop/alias_name.log;ddtdbg.ProtocolTraceShowTime=true`
	
	alias_name は JDBC 接続プールエイリアスの名前です。
	
	注:
	DB2 の場合は、値の末尾に以下の値を含めます。
	
	`ddtdbg.ProtocolTraceEBCDIC=true`
	
	通常、デフォルト値を変更する必要はありません。ただし、属性がシステムのニーズを満たさない場合は、値を変更することができます。この診断ツールは *Integration Server_directory* /instances/*instanceName*/logs/snoop ディレクトリからデータを収集することに注意してください。ログファイルの場所を変更すると、DataDirect Snoop ツールによって収集されるデータを診断ユーティリティがインポートできないことがあります。診断ユーティリティの使用方法の詳細については、995 ページの「診断ユーティリティの使用方法」を参照してください。DataDirect Snoop ツールの属性を設定する方法の詳細については、DataDirect Web サイトでマニュアルを参照してください。
	
	注:
	MySQL Community Edition はこのオプションをサポートしていません。

[ユーザ ID]	Integration Server がデータベースに接続するために使用するデータベースユーザ。
[パスワード]	Integration Server がデータベースに接続するために使用するパスワード。パスワードが必須ではない場合は、このフィールドを空白のままにします。
[接続数の最小値]	プールが常時オープンしておく必要のある最小の接続数。 複数のデータベースコンポーネントでこのプールを使用する場合、各プールインスタンスは指定された数の接続をオープンした状態に保ちます。たとえば、ISCoreAuditLog および DocumentHistory の両方のデータベースコンポーネントでこのツールを使用し、3 つ以上の接続をオープンした状態に保つよう指定した場合、プールは全体で 6 つの接続をオープンした状態に保ちます。つまり、ISCoreAuditLog プー

パラメータ	指定内容
	ルインスタンスに 3 つ、DocumentHistory プールインスタンスに 3 つです。 ログ記録量が急激に増加した場合、増加分の処理に必要な接続がオープンしていることをすぐに確認することで、パフォーマンスを改善することができます。この値を大きくすると、ログ急増中の接続の起動時間を最小限にし、常にオープンしている接続数を増やすことができます。
[接続数の最大値]	プールが一度にオープンできる最大の接続数。接続要求の数がこの値に達すると、Integration Server は要求をブロックします。 この値は、データベースに書き込みを行うすべての機能エイリアスおよびアプリケーションによって同時にオープン可能な合計の接続数の一部として計算します。合計数がデータベースの接続数の上限を超えないようにしてください。いずれかのアプリケーションが、データベースで許容されるより多くの接続をオープンすると、データベースはそれ以降あらゆるアプリケーションからの接続要求を拒否します。 Trading Networks もデータベースに書き込みを行い、最大 5 つの接続を開くプールを持っている場合、[**接続数の最小値**] で説明した例を続行するには、現在のプールに対して最大接続数は 17 しか指定できません。ISCoreAuditLog プールインスタンスは最大 17 の接続を使用でき、DocumentHistory プールインスタンスは残り 5 つの接続を使用できることになります。
[使用可能な接続の警告しきい値]	**[接続数の最大値]** のパーセンテージとして表され、常にプールで使用可能な接続数。 接続数がこの数に達しない場合や下回る場合、Integration Server がサーバログへのメッセージをログに記録します。接続数が後からこの数を上回った場合、Integration Server が接続プールのしきい値がクリアされたというサーバログへのメッセージをログに記録します。このしきい値を無効にする場合は、値を 0 に設定します。
[待機スレッドしきい値カウント]	一度に待機することができる最大接続要求数。 この数を超えると、Integration Server がサーバログへのメッセージを記録し、5 分の間隔タイマを起動します。間隔タイマの終了になっても要求数がこの数を超えている場合、Integration Server がサーバログへのメッセージをログに記録します。このしきい値を無効にする場合は、値を 0 に設定します。
[アイドルタイムアウト]	プールが未使用の接続をオープンしておくことのできる時間 (ミリ秒)。指定した時間が経過すると、[**接続数の最小値**] の値を満たすのには不要な、未使用の接続がクローズされます。デフォルトの有効期限は 60000 ミリ秒です。

4. [**接続テスト**] をクリックして、Integration Server がデータベースに接続できることを確認してください。

5. [**設定の保存**] をクリックします。

6. ISCoreAudit データベースコンポーネントの接続プールの値を確認します。[**ユーザ ID**] フィールドで指定したデータベースユーザが ISCoreAudit データベースコンポーネントを作成したデータベースユーザではない場合、watt.server.audit.schemaName プロパティを ISCoreAudit データベースコンポーネントを含むスキーマ名に設定します (1095 ページの「サーバ設定パラメータ」を参照)。

データベースへの送信接続にケルベロス認証を使用する

Integration Server では、JDBC 接続プールによるデータベースへの送信接続にケルベロス認証を使用できます。また、Integration Server はサポートされているデータベースでの DateDirect ドライバのケルベロス再認証機能の使用もサポートしています。DataDirect 接続パラメータ loginConfigName を使用することで、各 JDBC プールで異なるケルベロスチケットキャッシュを使用するよう設定できます。

ケルベロス再認証機能の使用

Integration Server は、サポートされているデータベースでの DateDirect ドライバのケルベロス再認証の使用をサポートしています。

ケルベロス認証を使用してデータベースに接続し、異なるデータベースユーザを使用してデータベースにアクセスするには

- 次の新しいデータベース URL パラメータを JDBC 接続プールに追加します。
  `ReAuthenticateUser=userName`

  たとえば、ケルベロスを使用して認証されたユーザの代わりに、「ISinternalUser」という名前の Oracle ユーザを使用してデータベースにアクセスするには、最初に必要なすべての Oracle 権限をそのユーザに譲渡し、次に JDBC プールのデータベース URL に次のプロパティを追加します。 `ReAuthenticateUser=ISinternalUser`

各 JDBC 接続プールで異なるケルベロスチケットキャッシュファイルを使用する

DataDirect 接続パラメータ loginConfigName を使用することで、各 JDBC プールで異なるケルベロスチケットキャッシュを使用するよう設定できます。

Integration Server の各 JDBC 接続プールで異なるケルベロスチケットキャッシュファイルを使用するには

1. *Integration Server_directory* /instances/*instanceName*/config フォルダにある is_jaas.cnf ファイルに新しいログインコンテキストを作成します。

2. 新しいログインコンテキスト内で、ログインコンテキストに使用するチケットキャッシュを指定します。

   次に例を示します。

   ```
   JDBC_ISInternal {
   com.sun.security.auth.module.Krb5LoginModule required
   useTicketCache="true"
   ticketCache="/opt/krb_tickets/isinternal.kd";
   };
   ```

3. `loginConfigName=<loginContext>` パラメータを指定して、JDBC 接続プールのデータベース URL にログインコンテキストを追加します。

   たとえば、is_jaas.cnf ファイルのログインコンテキストの名前が JDBC_ISInternal である場合、JDBC 接続プールのデータベース URL に次のパラメータを追加します。
   `loginConfigName=JDBC_ISInternal`

   DataDirect ドライバは JDBC_ISInternal JAAS ログインコンテキストにあるチケットキャッシュおよびその他の詳細情報を使用して、データベースに接続します。

ケルベロスの JAAS 競合の対処

別のコンポーネントとの JAAS 競合が発生すると、Integration Server JDBC プールはケルベロスを使用して正常にデータベースに接続できません。この問題が発生するのは、JDBC 接続が Oracle ojdbc8 ドライバを使用してケルベロス接続を確立した後です。Integration Server JDBC プールはケルベロスを使用してデータベースに接続できません。すべての接続の試行は失敗し、次のメッセージが表示されます。

`No LoginModules configured for loginConfigName from database URL.`

接続の試行が失敗するのは、Integration Server JAAS 設定が別のコンポーネントによって消去された JAAS 競合があるためです。

この問題に対処するには、JAAS をプログラムで実行して Subject を取得し、その Subject を使用してデータベースへの接続を確立するよう、Integration Server を設定します。この機能を使用するには、次の条件を満たす必要があります。

- サーバ設定パラメータ watt.server.jdbc.datadirect.useJaasSubjectForKerberos が「true」に設定されている必要があります。このパラメータは、JDBC プール接続でケルベロスを使用するときに Integration Server が JAAS 設定の Subject を使用することを指定します。このパラメータの変更を適用するために Integration Server を再起動する必要はありません。
- JDBC 接続プールエイリアスの [**データベース URL**] に `AuthenticationMethod=Kerberos` が含まれている必要があります。

10 ポートの設定

- ポートについて 169
- ポートの追加に関する考慮事項 172
- HTTP ポートの追加 174
- HTTPS ポートの追加 180
- ファイルポーリングポートについて 188
- FTPS ポートの追加 194
- FTP ポートの追加 200
- 電子メールポートの追加 203
- HTTP 診断ポートの追加 211
- HTTPS 診断ポートの追加 216
- HTTP/HTTPS ポートの一時停止 224
- HTTP/HTTPS ポートの再開 225
- HTTPS 要求のテスト 225
- FTP/FTPS ポート範囲の使用 225
- プライマリポートについて 227
- ポートの削除 228
- ポートの編集 228
- ポートの有効化/無効化について 228
- ポートによるクライアント認証の処理の設定 230

■ セキュリティプロバイダの追加 230
■ ポートごとに JSSE 用のプロトコルを無効にする 231
■ HTTP/S ポートをステートレスに指定する 233

ポートについて

Integration Server は指定ポートで要求を受信待機します。各ポートは、HTTP、HTTPS、FTP、FTPS、WebSocket、電子メール、ファイルポーリングなど、特定のプロトコルタイプに関連付けられます。Integration Server では、これらのポートタイプに加えて、診断用ポート、休止ポート、および webMethods Enterprise Gateway が使用する 2 つのポートも提供します。

設定可能なポートタイプ

次の表は、設定可能なポートタイプを示します。

使用するポートタイプ	目的	参照先
HTTP	セキュアでない要求をサーバにサブミットする。	174 ページの「HTTP ポートの追加」
HTTPS	SSL 暗号化を使用してサーバに要求をサブミットする。	180 ページの「HTTPS ポートの追加」
FTP	サーバとの間でファイルを移動する。	200 ページの「FTP ポートの追加」
FTPS	SSL 暗号化を使用してサーバとの間でファイルを移動する。	194 ページの「FTPS ポートの追加」
電子メール	POP3、IMAP などの電子メールサーバ経由で要求を受信する。	203 ページの「電子メールポートの追加」
ファイルポーリング	ファイルシステムが新しいファイルを受信したかどうかを監視して、受信時に特別な処理を行う。	189 ページの「ファイルポーリングポートの追加」
診断	サーバが無応答になったときに Integration Server Administrator にアクセスする。	211 ページの「HTTP 診断ポートの追加」
休止	サーバ保守のために休止モードを開始および終了する。	983 ページの「保守のためのサーバの休止」
Enterprise Gateway Server	外部クライアントからの要求を受信待機し、webMethods Enterprise Gateway 構成内の内部サーバへの接続を維持する。	593 ページの「webMethods Enterprise Gateway の設定 」
内部サーバ	内部サーバを webMethods Enterprise Gateway 構成内の Enterprise Gateway Server に接続する。	593 ページの「webMethods Enterprise Gateway の設定 」

使用するポートタイプ	目的	参照先
webMethods/WebSocket	WebSocket プロトコルを使用してサービスの要求をサブミットします。	861 ページの「WebSocket の設定」

デフォルトポート

デフォルトの Integration Server インスタンスには、次の事前設定済みのポートがあります。

エイリアス	プロトコル	ポートタイプ	ポート番号
DefaultPrimary	HTTP	プライマリ	5555
DefaultDiagnostic	HTTP	診断	9999
DefaultSecure	HTTPS	通常	5543

Integration Server でデフォルトポートに割り当てられたエイリアスは変更できません。デフォルトのポートとは異なるエイリアス、プロトコル、ポートタイプを使用する場合、このポートを削除して、新しいエイリアスでポートを再作成します。DefaultPrimary ポートに対して異なるエイリアスを使用するには、DefaultPrimary ポートを削除して新しいエイリアスで再作成する前に、別のポートをプライマリポートとして指定する必要があります。

同じマシン上で複数の Integration Server を実行している場合、各サーバのプライマリポート番号、診断ポート番号、および DefaultSecure ポート番号がそれぞれ固有である必要があります。複数の Integration Server インスタンスを実行する方法の詳細については、63 ページの「複数の Integration Server インスタンスの実行」を参照してください。

注:
DefaultSecure ポートでは、事前定義済みの DEFAULT_IS_KEYSTORE をキーストアエイリアスとして使用し、DEFAULT_IS_TRUSTSTORE をトラストストアエイリアスとして使用します (それぞれ、Integration Server のデフォルトのキーストアまたはトラストストアを含みます)。Software AG では、Integration Server のインストールが完了したら、すぐに DefaultSecure ポートで使用されるキーストアとトラストストアを変更することを強くお勧めします。変更しないと、Integration Server にインストールされたデフォルトのキーストアとトラストストアの情報を知る人物に対して、サーバが無防備な状態になります。

ポートエイリアスについて

ポートを作成する場合、ポートに対してエイリアスを指定します。ポートエイリアスは、ポートの一意の識別子として機能します。多くの場合、エイリアスはポートの目的を知らせる短い説明的な名前になります。たとえば、ポートエイリアスでポートの主な機能を識別できるようにすることができます (AdminPort、WebServicePort、projectNamePort など)。

Software AG Command Central を使用するときにポートエイリアスを利用すると、ポートの管理、設定、および比較が簡単になります。たとえば、Integration Server のグループですべての

管理ポートに同じ名前または類似の名前を割り当てた場合、容易に識別できるため、管理ポートの比較がより簡単になります。

ポートエイリアスは、次の条件を満たす必要があります。

- Integration Server で一意である。
- 1～255 文字の長さである。
- 文字 (a～z、A～Z)、数字、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する。

一度ポートを作成すると、エイリアスは変更できません。Integration Server で提供されるデフォルトのポートには、170 ページの「デフォルトポート」の指定に従って、事前定義済みのエイリアス名が付けられます。

ポートエイリアスを使用していなかった以前のバージョンから Integration Server 9.5 以降にアップグレードすると、次の命名規則に従って Integration Server は各ポートにエイリアスを割り当てます (電子メールポートおよびファイルポーリングポートを除く)。

*protocol*Listener_*portNumber_hostName_packageName*

protocol はポートに指定されたプロトコルです。*portNumber* は、ポートに割り当てられた番号であり、*hostName* はポートに指定されたホスト名で、*packageName* はポートが関連付けられるパッケージです。

電子メールポートの場合、Integration Server はポートエイリアスに対して次の命名規則を使用します。

EMailListener_*userName_hostName_packageName*

userName は、ポートに指定されたユーザ名です。hostName はポートに指定されたホスト名であり、packageName はポートが関連付けられるパッケージです。

ファイルポーリングポートの場合、Integration Server はポートエイリアスに対して次の命名規則を使用します。

FilePollingListener_*monitoringDirectory@contentType*

monitoringDirectory は Integration Server が新しいファイルを監視するディレクトリです。*contentType* はポートが処理するファイルのコンテンツタイプです。ファイルポーリングポートがコンテンツタイプを指定しない場合、Integration Server は ポートエイリアスから *contentType* を省略します。

以前のバージョンの Integration Server から展開したポート、および、以前のバージョンの Integration Server で作成されたパッケージでインストールされたポートに対して、Integration Server は上記の命名規則に従ってエイリアスを作成します。

注:
Integration Server がポートに割り当てるエイリアスは変更できません。

ポートへのパッケージの関連付け

ポートはすべてパッケージに関連付けられます。デフォルトで、ポートは WmRoot パッケージに関連付けられます。診断ポートを除く、すべてのポートタイプはアプリケーションパッケージと関連付けることができます。このように関連付けることで、パッケージを複製したときに、パッケージが新しいサーバで同じ番号のポートを継続して使用します。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。

ポートの割り当て先パッケージが無効になっている場合、ポートも無効になります。ポートは、パッケージのアクティブ化が成功すると再有効化されます。ポートが無効な間、ポートの有効化が成功するまで、これらのポート経由のサービスまたは他のリソースへのアクセスは失敗します。パッケージのホット展開の使用を予定している場合、ホット展開により更新されるパッケージにポートを関連付けない方がよいことがあります。このようにすると、Integration Server でホット展開中にパッケージがアンロードされたときにポートとリソースが利用できない状況を避けることができます。

重要:
パッケージに関連付けられているポートを設定する場合は注意が必要です。パッケージをターゲットサーバ上に複製した際、新しいポートが原因でシステムのセキュリティが低下する恐れがあります。たとえば、HTTP ポート 5556 に関連付けられているパッケージを複製する場合は、複製プロセスによって複製先のサーバに HTTP ポート 5556 が作成されます。セキュリティの高さを理由に複製先のサーバで通常 HTTPS ポートだけを使用している場合は、この新規ポートがサーバ上のセキュリティホールになる可能性があります。

ポートの追加に関する考慮事項

その他に、追加ポートを複数設定できます。追加ポートには、HTTP、HTTPS、FTP、FTPS、電子メール、ファイルポーリングなどのプロトコルのポートを関連付けることができます。

重要:
同じホストマシン上で複数の Integration Server インスタンスを実行している場合、各サーバのポートにそれぞれ固有のポート番号を設定する必要があります。

ポートを追加する理由

以下の理由により、ポートを追加する場合があります。

- 特定のポート番号を必要とするアプリケーションがある。
- 複数のタイプの受信待機プロトコルをサポートする必要がある。
- 同じプロトコルに対して複数のポートを開く必要がある。
- webMethods Enterprise Gateway 設定でサーバを展開する必要がある。この設定では、内部ファイアウォールの内側にあるサーバに渡す前に、DMZ 内の Enterprise Gateway Server が要求を取得します。Enterprise Gateway ポートの追加方法については、593 ページの「webMethods Enterprise Gateway の設定」を参照してください。

ポート設定に関する考慮事項

ポート設定に関する前提条件および考慮事項を以下に示します。

AS/400 に関する考慮事項

ポートキューは、TCP/IP スタックにキューイングされている未解決の受信接続の数を表します。サーバが AS/400 上で動作している場合、サーバプロパティ設定内に「watt.server.portQueue=511」という行を追加してポートキューのサイズを制限する必要があります。

サーバプロパティ設定を追加する方法については、146 ページの「拡張設定の使い方」を参照してください。

バインドアドレス

ポートを追加するときに、ポートにバインドする IP アドレスを指定できます。次の状況では、バインドアドレスを指定します。

- マシンに複数の IP アドレスが設定され、ポートで特定のアドレスを使用する。
- 複数のポートで同じポート番号を使用する。この場合、各ポートには異なるバインドアドレスを指定する必要があります。たとえば、ポート 7777 がバインドアドレスに関連付けられている場合、バインドアドレスを指定していない別のポート 7777 を追加することはできません。

バインドアドレスを指定しない場合、ポートはすべてのネットワークインタフェースを受信待機します。

SSL 用のポートを設定するための前提条件

HTTPS、FTPS または電子メールポートを設定する前に、SSL を使用してクライアント認証の妥当性検査に使用する認証を取得できるように、サーバを設定する必要があります。また、双方向の SSL 認証 (サーバもクライアントを認証する) の場合は、パートナアプリケーションの認証に Integration Server 認証マッピングが含まれている必要があります。

- **SSL を使用できるようにサーバを設定**SSL を使用できるようにサーバを設定する方法については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
- **CA の認証を取得**サーバがクライアント認証の妥当性検査に使用する、信用のあるルート認証です。これらの認証を取得する方法の 1 つに、Web ブラウザから入手するという方法もあります。ほとんどの場合、SSL をサポートする Web ブラウザでは、よく知られている認証局の認証が組み込まれています。認証が DER 形式であることを確認してください。DER 形式でない場合は、認証管理ツール (Java ***keytool*** など) を使用して DER 書式に変換してください。
- **パートナアプリケーションまたはリソースの認証マッピングを設定**Integration Server と情報を交換するパートナアプリケーションまたはリソースで SSL を使用した認証が必要な場合は、そのパートナまたはリソースに認証マッピングが含まれている必要があります。上記の

認証をトラストストアに保存し、Integration Server で使用する前にそのトラストストアのエイリアスを作成しておく必要があります。詳細については、を参照してください。

ポートの使用方法およびセキュリティ

セキュリティ上の理由から、5555 番以外の全ポートは、許可リストで指定されているサービス以外の全サービスへのアクセスを拒否するようにデフォルトで設定されています。必要に応じて、追加手順を実行して、そのポートで使用できるサービスを増やしてください。以下を参照してください。 230 ページの「ポートによるクライアント認証の処理の設定」

注:
UNIX システムで Integration Server が稼動している場合、1024 未満のポート番号を使用するにはサーバを root で実行する必要があります。Software AG ではセキュリティ上の理由からこの方法はお勧めしていません。代わりに、権限のないユーザ ID を使用して Integration Server を番号の大きいポート (1024 以上) で実行し、通常ファイアウォールに装備されているポート再マッピング機能を使用して番号の大きいポートに要求を移動します。

HTTP ポートの追加

HTTP ポートを追加するには、以下の手順に従います。

HTTP ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**webMethodsHTTP**] を選択します。
5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server Administrator のページが表示されます。以下の情報を指定します。

パラメータ	指定する値
[有効]	この HTTP リスナーを有効にするか ([**はい**]) 無効にするか ([**いいえ**]) を指定します。
[ポート]	ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバで使用されるポート番号がそれぞれ固有であることを確認してください。

パラメータ	指定する値
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[バインドアドレス(オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Integration Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 200 です。最大値は 65535 です。 **注:** [バックログ] パラメータは無効化されたポートには適用されません。Integration Server は、無効化されたポートへ送信する要求を拒否します。
[キープアライブタイムアウト]	タイムアウト値 (ミリ秒単位) に指定された時間内にサーバがクライアントから要求を受信しなかった場合に接続を終了するタイミング、または、クライアントがサーバに終了要求を明示的に送信した場合に接続を終了するタイミング。
[スレッドプール]	リスナーが要求のディスパッチにこのプールを排他的に使用するかどうか。Integration Server の既存のスレッドプールはグローバルスレッドプールです。リソースの負荷が非常に高い場合は、グローバルスレッドプールを通じて要求を処理するために長い待ち時間が発生することがあります。ただし、プライベートスレッドプールオプションを有効化すれば、このポートで受信した要求が他のサーバ機能とスレッドを競合することを回避できます。

パラメータ	指定する値
	このポートで受信した要求に関してプライベートスレッドプールを設定するには、[**有効**] をクリックします。以下のデフォルト設定をそのまま使用するか、変更することができます。 [**スレッドプールの最小値**] は、プライベートスレッドプールのスレッドの最小数を示します。デフォルトは 1 です。 [**スレッドプールの最大値**] は、プライベートスレッドプールのスレッドの最大数を示します。デフォルトは 5 です。 [**スレッドの優先順位**] は、Java スレッドの優先順位を示します。デフォルトは 5 です。 **重要:** この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。 スレッドプール機能を使用しない場合は、[**無効**] をクリックします。 ポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数がサーバから報告されます。

6. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定する値
[クライアントの認証]	この HTTP ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。次のいずれかを選択します。 **オプション** / **説明** **[ユーザ名/パスワード]** — Integration Server はクライアントに対してユーザ ID とパスワードを要求します。 **[Digest]** — Integration Server は、すべての要求を認証するために、パスワードダイジェストを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してダイジェストスキームで HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報

パラメータ	指定する値	
		を提供する場合、Integration Server は要求を検証および検査します。 クライアント要求の認証用にパスワードダイジェストを使用するように設定されたポートは、ユーザが認証にパスワードダイジェストを許可するように設定されている場合に限り、そのユーザからの要求を処理します。ダイジェスト認証用にユーザを設定する方法の詳細については、104 ページの「ユーザアカウントの追加」を参照してください。
	[ケルベロスチケットを要求する]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTP 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server は基本認証のユーザ名とパスワードを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。
	[ケルベロスチケットを必須にする]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTP 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server の認証は失敗します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。
[ケルベロスプロパティ] (オプション)	ケルベロスプロパティを使用して、ケルベロスチケットと共に受け取られるサービス要求の処理に使用するケルベロス関連の詳細を提供して、ケルベロス認証を有効にします。ケルベロス	

パラメータ	指定する値
	認証の設定の詳細については、552ページの「ケルベロス認証」を参照してください。

[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキストを指定します。
	
	以下の例で、**JAAS コンテキスト**は `KerberosClient` です。
	
	`KerberosClient {`
	`com.sun.security.auth.module.`
	`Krb5LoginModule required`
	`useKeyTab=true`
	`keyTab=alice.keytab;`
	`};`
	
	Integration Server で配布される is_jaas.cnf ファイルには、受信要求で使用できる IS_KERBEROS_INBOUND という名前の JAAS コンテキストが含まれます。

[プリンシパル]	ケルベロス認証で使用されるプリンシパルの名前を指定します。
[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワードを指定します。プリンシパルとそのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。 JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
[プリンシパルパスワードの再入力]	プリンシパルパスワードを再入力します。
[サービスプリンシパル名形式]	**[ユーザ名]** が表示されます。KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義さ

パラメータ	指定する値	
		れている名前付きユーザとしてサービスのプリンシパル名が表されることを示します。
	[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスで使用するプリンシパルの名前を指定します。**サービスプリンシパル名**は、以下の形式で指定します。 *principal-name.instance-name@realm-name*

7. [**変更内容の保存**] をクリックします。

8. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

   ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

9. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

HTTPポートの拡張コントロール

デフォルトでは、Integration Server はポート接続要求を受信すると直ちに要求を受け入れます。このため、ポートが同時に複数の要求を受信したときにこれらの要求を処理できるリソースがないと、問題が発生する可能性があります。[拡張コントロール] ページを使用して遅延値を指定することで、この問題を処理できます。遅延値が指定されている場合、Integration Server は指定された時間 (ミリ秒単位) だけ保留した後、ポートに対する接続要求を受け入れます。[拡張コントロール] ページを使用すれば、プライベートスレッドプールが有効になっている場合に、そのサイズを超えて、どの程度までリスナーが接続を受け入れるかを制御できます。

拡張コントロールの編集

拡張コントロールを編集するには、以下の手順に従います。

拡張コントロールを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

   ページのメイン領域に [**ポートの一覧**] テーブルが表示されます。

3. このテーブルの [**拡張**] 列の [**編集**] をクリックします。

   Integration Server によって、リスナーコントロールおよびプライベートスレッドプールコントロールに関する情報を要求するページが表示されます。このページの [診断 HTTP リスナーの状態] および [プライベートスレッドプール] 領域には事前定義された値が表示されています。

4. 次の情報を入力します。

パラメータ	指定する値
[**リスナーコントロール**]	プライベートスレッドプールが有効化されている場合に、リスナーによる接続の受け入れ方法およびその他のコントロールを管理するために設定するコントロールのタイプ。 [**一時停止**]。リスナーによる接続の受け入れを停止し、その後の要求のディスパッチを停止します。 [**増やす**]。リスナーが新規クライアント接続を受け入れるまで待つ時間を延長します。 [**減らす**]。リスナーが新規クライアント接続を受け入れるまで待つ時間を短縮します。 [**設定する (遅延時間 (ミリ秒))**]。遅延時間をミリ秒単位で設定します。
[**プライベートスレッドプールコントロール**]。	Integration Server が複数の接続を処理している場合に、ポートと他のサーバ機能との競合を回避するために設定するスレッドプールコントロールのタイプ。 [**増やす**]。プライベートスレッドプールのスレッドの数を増やします。 [**減らす**]。プライベートスレッドプールのスレッドの数を減らします。 [**設定する (スレッド)**]。プライベートスレッドプールのスレッドの数を設定します。

5. [**適用**] をクリックして変更を確定します。または、[**キャンセル**] をクリックします。

HTTPS ポートの追加

HTTPS ポートでは Integration Server がクライアントとサーバを安全に認証し、交換データを暗号化することができます。デフォルトでは、HTTPS リスナーはデフォルトの Integration Server SSL キーの認証を使用します。ただし、Integration Server キーストア内にあるリスナー自体の秘密鍵を使用するようにリスナーを設定できます (ファイルベースまたは SmartCard/HSM ベース)。詳細については、492 ページの「Integration Server の SSL キーと認証の設定」を参照してください。

さらに、サーバが実行するクライアント認証のタイプを設定することもできます。クライアント認証を使用すると、クライアントの ID を確認できます (詳細については、549 ページの「クライアントの認証」を参照してください)。

HTTPS ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**webMethodsHTTPS**] を選択します。
5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server Administrator のページが表示されます。
6. [**正規 HTTPS リスナー設定**] で、次の情報を入力します。

パラメータ	指定する値
[**有効**]	この HTTPS または FTPS リスナーを有効にするか ([**はい**]) 無効にするか ([**いいえ**]) を指定します。
[**ポート**]	ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバで使用されるポート番号がそれぞれ固有であることを確認してください。
[**エイリアス**]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[**説明**]	ポートの説明。
[**パッケージ名**]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に

パラメータ	指定する値
	便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[バインドアドレス(オプション)]	このポートをバインドするIPアドレス。バインドアドレスは、使用するコンピュータに複数のIPアドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Integration Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 200 です。最大値は 65535 です。 **注:** [バックログ] パラメータは無効化されたポートには適用されません。Integration Server は、無効化されたポートへ送信する要求を拒否します。
[キープアライブタイムアウト]	タイムアウト値 (ミリ秒単位) に指定された時間内にサーバがクライアントから要求を受信しなかった場合に接続を終了するタイミング、または、クライアントがサーバに終了要求を明示的に送信した場合に接続を終了するタイミング。
[スレッドプール]	リスナーが要求のディスパッチにこのプールを排他的に使用するかどうか。Integration Server の既存のスレッドプールはグローバルスレッドプールです。リソースの負荷が非常に高い場合は、グローバルスレッドプールを通じて要求を処理するために長い待ち時間が発生することがあります。ただし、プライベートスレッドプールオプションを有効化すれば、このポートで受信した要求が他のサーバ機能とスレッドを競合することを回避できます。 プライベートスレッドプールの設定を有効にするには、[**有効**] をクリックします。以下のデフォルト設定をそのまま使用するか、変更することができます。 [**スレッドプールの最小値**] は、プライベートスレッドプールのスレッドの最小数を示します。デフォルトは 1 です。 [**スレッドプールの最大値**] は、プライベートスレッドプールのスレッドの最大数を示します。デフォルトは 5 です。 [**スレッドの優先順位**] は、Java スレッドの優先順位を示します。デフォルトは 5 です。 **重要:** この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。

パラメータ	指定する値
	スレッドプール機能を使用しない場合は、[**無効**] をクリックします。 ポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数がサーバから報告されます。

7. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定する値
[JSSE の使用]	このポートで TLS 1.1、TLS 1.2、または TLS 1.3 をサポートする必要がある場合は、[**はい**] をクリックして、JSSE (Java Secure Socket Extension) ソケットファクトリを使用するポートを作成します。デフォルトは [**はい**] です。 この値を [**いいえ**] に設定すると、ポートはセキュアではない TLS 1.0 のみをサポートします。 **注:** JSSE を使用する Integration Server ポートで使用される暗号化スイートを制御し、受信要求を処理するには、watt.net.jsse.server.enabledCipherSuiteList を設定します。詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。
[クライアントの認証]	この HTTPS ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。次のいずれかを選択します。

オプション	説明
[ユーザ名/パスワード]	Integration Server はクライアントに対してユーザ ID とパスワードを要求します。
[Digest]	Integration Server は、すべての要求の認証に対してパスワードダイジェストを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してダイジェストスキームで HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。

パラメータ	指定する値
	注: クライアント要求の認証用にパスワードダイジェストを使用するように設定されたポートは、ユーザが認証にパスワードダイジェストを許可するように設定されている場合に限り、そのユーザからの要求を処理します。ダイジェスト認証用にユーザを設定する方法の詳細については、104 ページの「ユーザアカウントの追加」を参照してください。
[クライアント認証を要求する]	Integration Server はすべての要求に対してクライアント認証を要求します。クライアントから認証が提示されなかった場合、サーバはクライアントにユーザ ID とパスワードを要求します。クライアントから認証が提供された場合は、次の処理が実行されます。 ■ サーバは、認証がファイルにあるクライアント認証と正確に一致し、信用のある認証局によって署名されているかどうかをチェックします。上記のチェックに該当する場合、クライアントは Integration Server 内で認証がマッピングされているユーザとしてログインします。該当しない場合、セントラルユーザ管理が設定されていない限り、クライアント要求は失敗します。 ■ セントラルユーザ管理が設定されている場合、サーバは、セントラルユーザデータベース内で認証がユーザにマッピングされているかどうかをチェックします。マッピングされている場合、クライアントはそのユーザとしてサーバにログインします。マッピングされていない場合、クライアント要求は失敗します。
[クライアント認証を必須にする]	Integration Server はすべての要求に対してクライアント認証を必須にします。サーバは、クライアントが常に認

パラメータ	指定する値	
		証を提示する必要があることを除いて、[**クライアント認証を要求する**] を指定した場合と同様に動作します。クライアントがクライアント認証を提示しなければ、要求は失敗します。
	[ID プロバイダを使用]	Integration Server は OpenID プロバイダを使用して要求を認証します。Integration Server はこのポートに送信されるすべての要求を [**ID プロバイダ**]. に指定された OpenID プロバイダにリダイレクトします。
	[ケルベロスチケットを要求する]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTPS 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server は SSL ハンドシェークからクライアント証明書を使用して認証しようとします。クライアント証明書が見つからない場合は、Integration Server は基本認証のユーザ名とパスワードを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。
	[ケルベロスチケットを必須にする]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTPS 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server の認証は失敗します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。

パラメータ	指定する値
[ケルベロスプロパティ] (オプション)	ケルベロスプロパティを使用して、ケルベロスチケットと共に受け取られるサービス要求の処理に使用するケルベロス関連の詳細を提供して、ケルベロス認証を有効にします。ケルベロス認証の設定の詳細については、552ページの「ケルベロス認証」を参照してください。

[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキストを指定します。
	
	以下の例で、**JAAS コンテキスト**は `KerberosClient` です。
	`KerberosClient {`
	`com.sun.security.auth.module.`
	`Krb5LoginModule required`
	`useKeyTab=true`
	`keyTab=alice.keytab;`
	`};`
	
	Integration Server で配布される is_jaas.cnf ファイルには、受信要求で使用できる IS_KERBEROS_INBOUND という名前の JAAS コンテキストが含まれます。

[プリンシパル]	ケルベロス認証で使用されるプリンシパルの名前を指定します。
[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワードを指定します。プリンシパルとそのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。 JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
[プリンシパルパスワードの再入力]	プリンシパルパスワードを再入力します。

パラメータ	指定する値
[サービスプリンシパル名形式]	**[ユーザ名]** が表示されます。KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義されている名前付きユーザとしてサービスのプリンシパル名が表されることを示します。
[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスで使用するプリンシパルの名前を指定します。**サービスプリンシパル名**は、以下の形式で指定します。 *principal-name.instance-name@realm-name*

8. [**リスナー固有のクレデンシャル (オプション)**] で、次の情報を入力します。

注:
[認証] ページで指定した認証とは異なる認証セットを使用する場合にのみ、この設定を使用します。

パラメータ	指定する値
[キーストアエイリアス]	オプション。Integration Server キーストアのユーザ指定のテキスト識別子。 エイリアスは秘密鍵および関連する認証のリポジトリを指し示します。それぞれのリスナーで 1 つのキーストアを指し示しますが、同じキーストア内に複数のキーおよび認証が存在し、複数のリスナーで同じキーストアエイリアスを使用することもできます。 詳細については、513ページの「キーストアエイリアスの作成」を参照してください。
[キーエイリアス]	オプション。上記のキーストアエイリアスによって指定されるキーストアに保存されている必要がある、秘密鍵のエイリアス。
[トラストストアエイリアス]	オプション。トラストストアのエイリアス。トラストストアには、キーエイリアスに関連付けられている Integration Server 認証に署名した認証局の信用のあるルート認証が格納されている必要があります。また、トラストストアには、Integration Server が信用関係の妥当性検査を行うために使用する認証局の認証のリストも格納されます。

9. [**変更内容の保存**] をクリックします。

10. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

    ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

11. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

ファイルポーリングポートについて

ファイルポーリングポートを使用すると、ファイルを受信したかどうかについて監視ディレクトリを定期的にポーリングします。その後、Integration Server はそのファイルに対して特別なファイル処理サービスを実行します。ファイルポーリングポートと処理は次のように動作します。

1. Integration Server のファイルポーリングポートを使用すると、ファイルを受信したかどうかについて監視ディレクトリを定期的にポーリングします。各ファイルポーリングポートは、監視ディレクトリを 1 つだけポーリングします。

2. 新しいファイルを検出すると、Integration Server はファイルを作業ディレクトリに移動します。Integration Server は、ファイル名の一意性を保証するランダムな内部カウンタを含むようにファイル名を変更します。その結果、作業ディレクトリのファイル名は次の命名規則に準拠します: FilePolling.<counter>.<file_Name>

   注:
   作業ディレクトリ、完了ディレクトリ、およびエラーディレクトリのすべてのファイルは上記の命名規則に従います。

3. Integration Server は、ポートに指定されたファイル処理サービスを実行します。ファイルは、解析、変換、妥当性検査などのサービスが行われた後、ファイルシステムに書き込まれます。このような特別サービスは自分で作成し、このポーリングポートから呼び出し可能な唯一のサービスになるように設定します。

4. サービスがファイルの処理に成功したかどうかによって、次のいずれかが実行されます。

   - ファイルの処理が正常に完了した場合は、Integration Server はファイルを完了ディレクトリに移動します。
   - ファイルの処理がエラーで終了した場合は、Integration Server はファイルをエラーディレクトリに移動します。

5. Integration Server は、定期的な間隔で完了ディレクトリとエラーディレクトリをクリーンアップします。完了ディレクトリとエラーディレクトリのファイルが削除されずに保持される期間と、Integration Server が完了ディレクトリとエラーディレクトリをクリーンアップする頻度は、設定できます。

注:
作業ディレクトリのファイルの処理が完了する前に Integration Server がシャットダウンするまたは使用不可能になると、再起動後、ファイルは作業ディレクトリに残っています。Integration Server は、作業ディレクトリに残っているファイルの処理を再起動または再開し

ません。管理者は、ファイルを作業ディレクトリから監視ディレクトリに移行したり、ファイルを削除するかどうかを決定する必要があります。

Integration Server のクラスタまたは Integration Server の非クラスタグループでも、ファイルのポーリングは個別の Integration Server の場合とほぼ同じように作業されます。唯一の違いは、複数の Integration Server が監視ディレクトリをポーリングすることです。グループの Integration Server が監視ディレクトリからファイルを取得すると、グループの他の Integration Server はファイルを使用できなくなります。

ファイルのポーリングを設定するには、以下の作業を実行する必要があります。

- Integration Server で監視ディレクトリを設定します。ファイルポーリングに使用する他のディレクトリは Integration Server によって自動的に作成されます。
- ファイル処理サービスを作成し、Integration Server で使用できるようにします。ファイル処理サービスの例については、『*webMethods Service Development Help*』および『*Flat File Schema Developer's Guide*』を参照してください。
- Integration Server でファイルポーリングポートを設定します。

ファイルポーリングポートの追加

以下の手順に従ってファイルポーリングポートを Integration Server に追加します。

ファイルポーリングポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**webMethodsFilePolling**] を選択します。
5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server Administrator のページが表示されます。
6. [パッケージ] で、以下の情報を入力します。

パラメータ	指定する値
[パッケージ名]	このポートに関連付けるパッケージ。 パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。特別なファイル処理を実行する場合、その処理を実行するサービスが含まれているパッケージを指定します。このポートでフラットファイルを処理する場合は、[**WmFlatFile**] を選択します。このパッケージには、

パラメータ	指定する値
	フラットファイルの処理に使用可能な組み込みサービスが含まれています。 **注:** サービスのパッケージを複製する場合、サーバが同じコンピュータ上に存在しているか、または異なるコンピュータ上に存在しているかにかかわらず、同じ設定のファイルポーリングポートがターゲットサーバ上に作成されます。ファイルポーリングポートが既にターゲットサーバに存在している場合、ポートの設定は元のままになります。元のサーバとターゲットサーバが同じコンピュータにインストールされている場合、同じ監視ディレクトリが共有されます。ターゲットサーバが異なるコンピュータにインストールされている場合、デフォルトでは、別の監視ディレクトリがターゲットサーバのコンピュータ上に作成されます。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。

7. [**ポーリング情報**] で、次の情報を入力します。

パラメータ	指定する値
[監視ディレクトリ]	ファイルの監視に使用する Integration Server 上のディレクトリ。
[作業ディレクトリ (オプション)]	[**監視ディレクトリ**] で識別されたファイルを処理するための移動先となる Integration Server 上のディレクトリ。**作業ディレクトリ**に移動されるファイルは、経過日数とファイル名の要件を満たしている必要があります。ディレクトリを指定しなかった場合、*MonitoringDirectory*\..\Work というデフォルトのサブディレクトリが自動的に作成されます。
[完了ディレクトリ (オプション)]	[**監視ディレクトリ**] または [**作業ディレクトリ**] で処理が完了したファイルの移動先となる Integration Server 上のディレクトリ。ディレクトリを指定しなかった場合、*MonitoringDirectory*\..\Done というデフォルトのサブディレクトリが自動的に作成されます。 **注:** ファイルにタイムスタンプを付加してから [**完了ディレクトリ**] に移動するように Integration Server を設定できます。そのためには、appendTimeStampToFileName=true を

パラメータ	指定する値
	含むように properties.cnf ファイルを更新します。ファイル名は次の規則に従います: FilePolling.<timestamp>.<counter>.<file_Name> properties.cnf ファイルは、以下の場所にあります。 *Integration Server_directory* \instances*instanceName*\packages\WmFlatFile\config properties.cnf ファイルが存在しない場合は作成します。
[**エラーディレクトリ (オプション)**]	処理に失敗したファイルの移動先となる Integration Server 上のディレクトリ。ディレクトリを指定しなかった場合、*MonitoringDirectory*\..\Error というデフォルトのサブディレクトリが自動的に作成されます。
[**ファイル名フィルタ (オプション)**]	**監視ディレクトリ**のファイルに適用するファイル名フィルタ。Integration Server は、このフィルタ要件を満たすファイルのみ処理します。このフィールドを指定しなかった場合、すべてのファイルがポーリングされます。このフィールドでパターンマッチングを指定することもできます。
[**ファイル経過時間 (オプション) (秒)**]	**監視ディレクトリ**のファイルの処理が開始されるまでの最小経過時間 (秒単位)。Integration Server が指定するファイル経過時間は、ファイルが監視ディレクトリで最後に変更された時点に基づいています。監視ディレクトリへのファイル全体のコピーが完了する前にサーバで処理が実行されないよう、必要に応じてこの経過時間を調整することができます。デフォルトは 0 です。
[**コンテンツタイプ**]	ファイルの処理に使用するコンテンツタイプ。このフィールドで指定したコンテンツタイプに関連付けられているコンテンツハンドラが使用されます。この値を指定しなかった場合、Integration Server はファイル拡張子に基づいて MIME マッピングを実行します。 ファイルポーリングポートでは XML コンテンツを次のように処理します。 ■ コンテンツタイプが text/xml である場合、コンテンツは 'node' {com.wm.lang.xml.Document} としてパイプラインに渡されます。 ■ コンテンツタイプが設定されていない場合、text/xml のデフォルトの MIME マッピングが使用されます。この場合も、XML コンテンツは 'node' {com.wm.lang.xml.Document} としてパイプラインに渡されます。

パラメータ	指定する値
	■ コンテンツタイプが application/xml である場合、XML コンテンツは 'ffdata' {`java.io.BufferedInputStream`} としてパイプラインに渡されます。 **注:** デフォルトのエンコーディングを上書きするには、properties.cnf ファイルに filepollingport=*encodingType* を含めて、*encodingType* をフラットファイルパーサが XML ファイルの解析のために使用するエンコーディングとして更新します。 properties.cnf ファイルは、以下の場所にあります。 *Integration Server_directory* \instances*instanceName*\packages\WmFlatFile\config properties.cnf ファイルが存在しない場合は作成します。
[再帰的ポーリングを許可]	Integration Server で [**監視ディレクトリ**] 内のすべてのサブディレクトリをポーリングするかどうかを指定します。[**はい**] または [**いいえ**] を選択します。
[クラスタリングを有効にする]	Integration Server で監視ディレクトリでのクラスタリングを許可するかどうか。[**はい**] または [**いいえ**] を選択します。
[ロックファイルの拡張子 (オプション)]	特定の拡張子に対するポーリングを定義します。デフォルトは「lock」です。
[間隔ごとに処理されるファイル数 (オプション)]	間隔ごとにファイルポーリングリスナーで処理できるファイルの最大数を指定します。正の整数を指定した場合、監視ディレクトリからその数のファイルのみがファイルポーリングリスナーで処理されます。監視ディレクトリに残ったファイルは、後続の間隔で処理されます。-1 を指定するか値を指定しない場合、リスナーが監視ディレクトリのすべてのファイルを処理します。デフォルトは -1 です。

8. [**セキュリティ**] の下にある [**ユーザを指定してサービスを実行**] パラメータで、ファイルポーリングディレクトリに割り当てられているサービスの実行に使用するユーザ名を指定します。 をクリックしてユーザを検索して選択します。内部ユーザでも外部ユーザでも使用できます。

9. [**メッセージの処理**] で、次の情報を入力します。

パラメータ	指定する値
[有効]	このファイルポーリングポートを有効にするか ([**はい**]) 無効にするか ([**いいえ**])。

パラメータ	指定する値
[処理サービス]	ポーリングされたファイルに対して実行するサービスの名前。作業ディレクトリへのファイルのコピーが完了すると、このサービスが実行されます。このサービスがこのファイルポーリングポートから使用可能な唯一のサービスになるように設定する必要があります。 **重要:** ファイルポーリングポートで使用する処理サービスを変更した場合、そのポートから使用可能なサービスのリストにこの新しいサービスが含まれるように変更してください。詳細については、以下の項目を参照してください。
[ファイルポーリングの間隔(秒)]	Integration Server で **[監視ディレクトリ]** 内のファイルをポーリングする頻度 (秒単位)。
[ディレクトリの可用性が変更になったときのみログ]	**[いいえ]** (デフォルト) を選択した場合、リスナーは監視ディレクトリが使用不可になるたびにログにメッセージを記録します。 **[はい]** を選択した場合、リスナーは次の状況でログにメッセージを記録します。 ■ ディレクトリは直前のポーリングの際に使用可能であったが、今回のポーリング試行では使用不可である。 ■ ディレクトリは直前のポーリングの際に使用不可であったが、今回のポーリング試行では使用可能である。
[ディレクトリが NFS マウントファイルシステム]	UNIX システムで、監視ディレクトリ、作業ディレクトリ、完了ディレクトリ、およびエラーディレクトリがローカルファイルシステムにマウントされたネットワークドライブである場合に使用します。 **[いいえ]** (デフォルト) を選択した場合、リスナーは Java の File.renameTo() メソッドを呼び出して、監視ディレクトリから作業ディレクトリに、そして作業ディレクトリから完了ディレクトリおよびエラーディレクトリにファイルを移動します。 **[はい]** を選択した場合、リスナーは最初に監視ディレクトリからファイルを移動するために、Java の File.renameTo() メソッドを呼び出します。このメソッドが失敗した場合、リスナーは監視ディレクトリから作業ディレクトリにファイルをコピーし、監視ディレクトリからファイルを削除します。コピーアクションまたは削除アクションが失敗した場合、この操作は失敗します。作業ディレクトリから完了ディレクトリおよびエラーディレクトリにファイルを移動する際にも、同じ動作が適用されます。

パラメータ	指定する値
[クリーンアップサービス (オプション)]	[**ポーリング情報**] で指定されているディレクトリをクリーンアップするサービスの名前。
[起動時にクリーンアップ]	ファイルポーリングポートの起動時に**完了ディレクトリ**と**エラーディレクトリ**にあるファイルをクリーンアップするかどうか。
[クリーンアップのファイル経過時間 (オプション) (日)]	ディレクトリから削除する処理済みファイルの経過日数。デフォルトは 7 日に設定されています。
[クリーンアップの間隔 (オプション) (時間)]	Integration Server で処理済みファイルのクリーンアップをチェックする頻度 (時間単位)。デフォルトでは 24 時間に設定されています。
[呼び出しの最大回数]	Integration Server でポートに対して使用するスレッドの数。1 以上 10 以下の値を指定できます。

10. [**変更内容の保存**] をクリックします。

11. ポートのアクセスモードを適切に設定し、指定したファイル処理サービスがこのポートからアクセス可能な唯一のサービスになるように設定するには、以下の手順に従います。

    a. [**ポート**] ページで、作成したポートの [**アクセスモード**] フィールドの [**編集**] をクリックします。

    b. [**アクセスモードのデフォルトを拒否に設定**] をクリックします。

    c. [**許可リストへのフォルダとサービスの追加**] をクリックします。

    d. [**1 行につき 1 フォルダまたは 1 サービスを記述します。**] の下にあるテキストボックスに、目的の処理サービスの名前を入力します。

    e. 許可リストから他のサービスをすべて削除します。

    f. [**追加の保存**] をクリックします。

    注:
    ファイルポーリングポートで使用する処理サービスを変更した場合、そのポート用の許可リストも変更する必要があります。許可サービスのリストを変更するには、上記の手順に従ってください。

FTPS ポートの追加

FTPS (FTP over SSL) ポートでは、サーバが FTP クライアントおよびサーバを安全な方法で認証し、FTP クライアントとサーバ間で制御とデータの交換を暗号化することができます。

FTPS ポートを追加する場合は、以下の点に留意してください。

- FTPS クライアントは常にユーザ ID およびパスワードを要求されます。
- デフォルトで、FTPS ポートはセキュアなクライアントに対してだけ機能します。セキュアなクライアントとは、AUTH コマンドの発行によって接続のセキュリティを確保するクライアントです。FTPS リスナーをセキュアでないクライアントに対して機能するように設定することもできます。
- FTPS ポートは、それ自体の認証を使用するように設定するか、Integration Server 認証を使用する、またはクライアントの認証を要求する (または必要とする) ように設定できます。さらに、キーストア内の秘密鍵および認証チェーンを使用するようにリスナーを設定できます (ファイルベースまたは SmartCard/HSM ベース)。クライアント認証の詳細については、549 ページの「クライアントの認証」を参照してください。
- デフォルトでは、Integration Server は FTPS ポートの認証マッピングを実行しません。この機能を使用するには、watt.net.ftpUseCertMap 設定プロパティを「true」に設定する必要があります。FTPS ポートでのクライアント認証の仕組みの詳細については、199 ページの「FTPS ポートでのクライアント認証の仕組み」を参照してください。認証のマッピングの詳細については、553 ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。
- ユーザが FTPS ポートを介してログインすると、そのユーザは Integration Server によってデフォルトの FTP ルートディレクトリまたはクライアントユーザディレクトリに配置されます。Integration Server は、watt.server.login.userFtpDir パラメータの設定に基づいてディレクトリを選択します。詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

FTPS ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**webMethodsFTPS**] を選択します。
5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server のページが表示されます。次の情報を入力します。

パラメータ	指定する値
[有効]	この FTPS ポートを有効にするか ([**はい**]) 無効にするか ([**いいえ**]) を選択します。
[ポート]	ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。

パラメータ	指定する値
	重要: 同じホストマシン上で複数の Integration Server を実行している場合、各サーバで使用されるポート番号がそれぞれ固有であることを確認してください。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[バインドアドレス (オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[パッシブモードリスンアドレス (オプション)]	PORT コマンドによって送信されるアドレス。ホスト名または IP アドレスを指定できます。 **注:** このオプションは、FTPS ポートが IPv6 アドレスにバインドされている場合は適用されません。その場合、パッシブモードリスンアドレスはポートバインドアドレスと同じです。 パッシブモードで実行した場合、FTPS ポートは FTPS クライアントに PORT コマンドを送信します。PORT コマンドは、クライアントがデータ接続を作成するための接続先アドレスおよびポートを指定します。ただし、FTPS ポートが NAT サーバの内側にある場合、Integration Server が実行されているホストのアドレスは FTPS クライアントからは見えません。このため、PORT コマンドには、クライアントがサーバへの接続に必要とする情報が含まれません。この問題は、*Integration Server_directory* \instances*instance_name*\config

パラメータ	指定する値
	ディレクトリにあるサーバ設定ファイル (server.cnf) の中の watt.net.ftpPassiveLocalAddr プロパティの値を指定することで解決できます (1095 ページの「サーバ設定パラメータ」を参照)。 また、[**パッシブモードリスンアドレス**] フィールドを使用して、個々の FTPS ポートのパッシブモードアドレスを指定することもできます。この方法を使用すれば、FTPS ポートごとに異なるパッシブモードアドレスを指定できます。[**パッシブモードリスンアドレス**] フィールドと watt.net.ftpPassiveLocalAddr プロパティの両方にアドレスが指定されている場合、PORT コマンドは、watt.net.ftpPassiveLocalAddr プロパティに指定された値を使用します。
[セキュアクライアントのみ]	FTPS リスナーがセキュアでないクライアントに対して機能するのを防ぐ場合は、このチェックボックスをオンにします。

6. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定する値
[JSSE の使用]	このポートで TLS 1.1、TLS 1.2、または TLS 1.3 をサポートする必要がある場合は、[**はい**] をクリックして、JSSE (Java Secure Socket Extension) ライブラリを使用するポートを作成します。デフォルトは [**はい**] です。 この値を [**いいえ**] に設定すると、ポートはセキュアではない TLS 1.0 のみをサポートします。送信 FTPS 接続の作成には、Entrust IAIK ライブラリが使用されます。 **注:** JSSE を使用する Integration Server ポートで使用される暗号化スイートを制御し、受信要求を処理するには、watt.net.jsse.server.enabledCipherSuiteList を設定します。詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。
[クライアントの認証]	この FTPS ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。次のいずれかを選択します。

オプション	説明
[ユーザ名/パスワード]	Integration Server はクライアントに対してユーザ ID とパスワードを要求します。

パラメータ	指定する値	
	[クライアント認証を要求する]	Integration Server はすべての要求に対してクライアント認証を要求します。クライアントから認証が提示されなかった場合、サーバはクライアントにユーザ ID とパスワードを要求します。
	[クライアント認証を必須にする]	Integration Server はすべての要求に対してクライアント認証を必須にします。サーバは、クライアントが常に認証を提示する必要があることを除いて、[**クライアント認証を要求する**] を指定した場合と同様に動作します。

注:
watt.net.ftpUseCertMap サーバ設定パラメータの値に従って、Integration Server がどのように FTPS ポートで受信されるクライアント要求を処理するかに関する詳細については、199 ページの「FTPS ポートでのクライアント認証の仕組み」を参照してください。

7. [**リスナー固有のクレデンシャル (オプション)**] で、次の情報を入力します。

注:
[認証] ページで指定した認証とは異なる認証セットを使用する場合にのみ、この設定を使用します。

パラメータ	指定する値
[キーストアエイリアス]	オプション。Integration Server キーストアのユーザ指定のテキスト識別子。 エイリアスは秘密鍵および関連する認証のリポジトリを指し示します。それぞれのリスナーで 1 つのキーストアを指し示しますが、同じキーストア内に複数のキーおよび認証が存在し、複数のリスナーで同じキーストアエイリアスを使用することもできます。 詳細については、513 ページの「キーストアエイリアスの作成」を参照してください。
[キーエイリアス]	オプション。上記のキーストアエイリアスによって指定されるキーストアに保存されている必要がある、秘密鍵のエイリアス。
[トラストストアエイリアス]	オプション。トラストストアのエイリアス。トラストストアには、キーエイリアスに関連付けられている Integration

パラメータ	指定する値
	Server 認証に署名した認証局の信用のあるルート認証が格納されている必要があります。また、トラストストアには、Integration Server が信用関係の妥当性検査を行うために使用する認証局の認証のリストも格納されます。

8. [**変更内容の保存**] をクリックします。

9. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

   ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

10. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

FTPS ポートでのクライアント認証の仕組み

Integration Server が FTPS ポートで受信されたクライアント要求を処理する方法は、watt.net.ftpUseCertMap サーバ設定パラメータの値および FTPS ポートに対して設定されたクライアント認証によって異なります。

watt.net.ftpUseCertMap パラメータは、FTPS ポートで受信した要求に関して、Integration Server が認証マップを尊重するかどうかを指定します。

- 「false」に設定すると (デフォルト設定)、Integration Server はクライアントの認証に指定されたユーザを無視して、ユーザ名/パスワードのプロンプトを通じて入力された情報を使用してユーザをログインします。
- 「`true`」に設定した場合、クライアントの認証が事前に Integration Server ユーザにマッピングされていれば、Integration Server はクライアントの認証に指定されたユーザ名としてユーザをログインします。Integration Server は、ユーザ名/パスワードのプロンプトから入力されたユーザ名を無視します。

次の表では、watt.net.ftpUseCertMap が「true」に設定された場合に、Integration Server が、クライアント認証の設定と提供されている認証に基づいて、クライアント要求を処理する方法を説明します。

クライアント認証	提供されている認証	認証なし
[ユーザ名/パスワード]	プロンプトから入力されたユーザ名/パスワードでログインします。	プロンプトから入力されたユーザ名/パスワードでログインします。

クライアント認証	提供されている認証	認証なし
[クライアント認証を要求する]	認証が信用できるものであり、マッピングされたユーザに一致する場合は、そのユーザとしてログインします。 認証が信用できない、またはマッピングされたユーザと一致しない場合は、プロンプトから入力されたユーザ/パスワードでログインします。	プロンプトから入力されたユーザ名/パスワードでログインします。
[クライアント認証を必須にする]	認証が信用できるものであり、マッピングされたユーザに一致する場合は、そのユーザとしてログインします。プロンプトから入力されたユーザ/パスワードを無視します。 認証が信用できないか、マッピングされているユーザと一致しない場合は、プロンプトから入力されたユーザ/パスワードを無視し、ログイン要求を拒否します。	ログイン要求を拒否します。

次の表では、watt.net.ftpUseCertMap が「false」に設定された場合に、Integration Server が、クライアント認証の設定と提供されている認証に基づいて、クライアント要求を処理する方法を説明します。

クライアント認証	認証あり	認証なし
[ユーザ名/パスワード]	プロンプトから入力されたユーザ名/パスワードでログインします。	プロンプトから入力されたユーザ名/パスワードでログインします。
[クライアント認証を要求する]	認証が信用できる場合は受け入れますが、認証から提示されたユーザを無視します。代わりに、プロンプトから入力されたユーザ/パスワードでログインします。	プロンプトから入力されたユーザ名/パスワードでログインします。
[クライアント認証を必須にする]	認証が信用できる場合は受け入れますが、認証から提示されたユーザを無視します。代わりに、プロンプトから入力されたユーザ/パスワードでログインします。	ログイン要求を拒否します。

FTP ポートの追加

FTP ポートを使用すると、Integration Server との間でファイルを移動できます。FTP ポートを設定するには、以下の手順に従います。

ユーザが FTP ポートを介してログインすると、そのユーザは Integration Server によってデフォルトの FTP ルートディレクトリまたはクライアントユーザディレクトリに配置されます。Integration Server は、watt.server.login.userFtpDir パラメータの設定に基づいてディレクトリを選択します。詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

FTP ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**webMethodsFTP**] を選択します。
5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server のページが表示されます。次の情報を入力します。

パラメータ	指定する値
[ポート]	ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバで使用されるポート番号がそれぞれ固有であることを確認してください。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[バインドアドレス (オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されて

パラメータ	指定する値
	おり、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[パッシブモードリスンアドレス (オプション)]	PORT コマンドによって送信されるアドレス。ホスト名または IP アドレスを指定できます。 **注:** このオプションは、FTP ポートが IPv6 アドレスにバインドされている場合は適用されません。その場合、パッシブモードリスンアドレスはポートバインドアドレスと同じです。 パッシブモードで実行した場合、FTP ポートは FTP クライアントに PORT コマンドを送信します。PORT コマンドは、クライアントがデータ接続を作成するための接続先アドレスおよびポートを指定します。ただし、FTP ポートが NAT サーバの内側にある場合、Integration Server が実行されているホストのアドレスは FTP クライアントからは見えません。このため、PORT コマンドには、クライアントがサーバへの接続に必要とする情報が含まれません。この問題は、*Integration Server_directory* \instances*instance_name*\config ディレクトリにあるサーバ設定ファイル (server.cnf) の中の watt.net.ftpPassiveLocalAddr プロパティの値を指定することで解決できます (1095 ページの「サーバ設定パラメータ」を参照)。 また、[**パッシブモードリスンアドレス**] フィールドを使用して、個々の FTP ポートのパッシブモードアドレスを指定することもできます。この方法を使用すれば、FTP ポートごとに異なるパッシブモードアドレスを指定できます。[**パッシブモードリスンアドレス**] フィールドと watt.net.ftpPassiveLocalAddr プロパティの両方にアドレスが指定されている場合、PORT コマンドは、watt.net.ftpPassiveLocalAddr プロパティに指定された値を使用します。

6. [**変更内容の保存**] をクリックします。

7. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

   ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

8. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

電子メールポートの追加

Integration Server に 1 つ以上の電子メールポートを設定することにより、電子メールサーバ (POP3 または IMAP) を介してクライアント要求を受信できます。クライアントでは、実行するサービスの名前とそのサービスに渡されるパラメータを記載した電子メールを作成します。電子メールには、ユーザ ID およびパスワードに関する情報も記載できます。

電子メールポートを追加する前に、211 ページの「電子メールポートのセキュリティに関する考慮事項」で情報を確認します。

電子メールポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**webMethodsE-mail**] を選択します。
5. [**サブミット**] をクリックします。Integration Server にポート情報の入力を要求する [**電子メールクライアント設定の編集**] ページが表示されます。
6. [パッケージ] で、以下の情報を指定します。

パラメータ	指定する値
[**パッケージ名**]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[**エイリアス**]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[**説明**]	ポートの説明。

7. [**サーバ情報**] で、次の情報を入力します。

パラメータ	指定する値
[**有効**]	電子メールポートを有効にする ([**はい**]) か無効にするか ([**いいえ**])。ポートが有効になっている場合にのみ要求を受信できます。
[**タイプ**]	メールサーバのタイプ。[**POP3**] または [**IMAP**] を選択します。
[**ホスト名**]	POP3 サーバまたは IMAP サーバが稼動しているコンピュータ名。
[**ポート**]	Integration Server が接続される電子メールサーバのポート。 POP3 メールサーバの場合、明示的 SSL のデフォルトは 110 で、暗黙的 SSL のデフォルトは 995 です。 IMAP メールサーバの場合、明示的 SSL のデフォルトは 143 で、暗黙的 SSL のデフォルトは 993 です。
[**ユーザ名**]	電子メールサーバで識別するユーザ名。
[**認証**]	Integration Server が指定された電子メールサーバに接続するために使用する認証のタイプ。[**基本認証**] または [**OAuth**] を選択します。 ■ ログインクレデンシャル (ユーザ名とパスワード) のみ使用して特定の電子メールサーバでユーザを認証する場合は、[**基本認証**] を選択します。 ■ OAuth サーバで発行されたクレデンシャルを使用して特定の電子メールサーバでユーザを認証する場合は、[**OAuth**] を選択します。 **注:** この認証タイプは POP3 サーバに関連付けられた電子メールポート向けではありません。 **注:** OAuth には、認証コードの譲渡タイプのみサポートされています。 **注:** Integration Server 用の OAuth クレデンシャルを取得するには、最初に OAuth サーバに Integration Server を登録する必要があります。アプリケーションの登録および OAuth クレデンシャルの取得の詳細については、認証サーバのマニュアルを参照してください。

パラメータ	指定する値
[パスワード]	電子メールサーバで識別されるユーザ名に関連付けられたパスワード。このフィールドは基本認証にのみ適用されます。
[認証 URL]	Integration Server が認証コードの要求に使用する必要があるエンドポイントの URL。このフィールドは OAuth にのみ適用されます。
[クライアント ID]	OAuth サーバで登録時に Integration Server 用に生成される一意の公開識別子。このフィールドは OAuth にのみ適用されます。
[クライアントシークレット]	OAuth サーバで登録時に Integration Server に提供される一意の文字列。Integration Server および OAuth サーバでのみ認識されます。このフィールドは OAuth にのみ適用されます。
[スコープ]	登録時に Integration Server 用に設定された電子メールサーバへのアクセス権限。複数のスコープはスペースで区切って指定できます。このフィールドは OAuth にのみ適用されます。
[アクセストークン URL]	Integration Server が OAuth サーバからアクセストークンを要求するときに使用する必要があるエンドポイントの URL。このフィールドは OAuth にのみ適用されます。
[リダイレクト URL]	OAuth サーバが Integration Server に認証応答を送信するときに使用する必要がある URL。 ■ Integration Server にローカルにアクセスしている場合、次のいずれかの適用可能な形式でリダイレクト URL を入力します。 `http://localhost:{port}/WmRoot/security-oauth-get-authcode.dsp` または `https://localhost:{port}/WmRoot/security-oauth-get-authcode.dsp` ■ Integration Server にリモートにアクセスしている場合、`https://{ISHostName}:{port}/WmRoot/security-oauth-get-authcode.dsp` を使用します。 このフィールドは OAuth にのみ適用されます。
[認証コード]	Integration Server でアクセストークンの要求に必要な一意の文字列を OAuth サーバから受信するには、[**認証コードの取得**] をクリックします。 **注:** 電子メールポートの OAuth クレデンシャルを編集する場合は、必ず認証コードを取得してください。
[アクセストークンの有効期限]	webMethods Integration Server に発行されるアクセストークンが有効である期限を指定します。このフィールドは OAuth にの

パラメータ	指定する値
	み適用されます。これは読み取り専用フィールドであり、アクセストークンに基づいて生成されます。
[転送レイヤセキュリティ]	電子メールサーバとの通信時に Integration Server が使用する SSL 暗号化のタイプ。明示的または暗黙的な転送レイヤセキュリティを使用するか、または転送レイヤセキュリティを使用しないように、ポートを設定できます。

指定する値	目的
[なし]	Default電子メールサーバとの通信時にセキュアでないモードを使用します。 [**なし**] を指定した場合、電子メールサーバはユーザ名とパスワードのみを使用して電子メールクライアントを認証します。
[明示的]	電子メールサーバとの通信時に明示的なセキュリティを使用します。明示的なセキュリティの場合、Integration Server は電子メールサーバへの非暗号化接続を確立してから、セキュアモードにアップグレードします。 明示的な転送レイヤセキュリティでは、Integration Server は、SSL 暗号化をサポートしている電子メールサーバおよび SSL 暗号化をサポートしていない電子メールサーバと通信できます。電子メールサーバで転送レイヤセキュリティがサポートされていない場合、Integration Server は確立された電子メールサーバとの接続を切断します。[**転送レイヤセキュリティ**] フィールドで [**なし**] オプションを選択してポートを有効にすると、電子メールサーバに対してセキュアでない接続を確立することができます。
[暗黙的]	電子メールサーバとの通信時に暗黙的なセキュリティを使用します。暗黙的なセキュリティの場合、Integration Server は常に電子メールサーバに対して暗号化された接続を確立します。SSL をサポートしているクライアントのみアクセスを許可されます。

[トラストストアエイリアス (オプション)]	オプション。電子メールサーバによって Integration Server に提示される認証が含まれているトラストストアのエイリアス。トラストストアエイリアスを選択しなかった場合、watt.security.trustStoreAlias プロパティに指定されているデフォルトのトラストストアエイリアスが使用されます。このプロパティの詳細については、1129 ページの「watt.security.」を参照してください。トラストストアエイリアスの詳細については、

パラメータ	指定する値
	515ページの「トラストストアエイリアスの作成」を参照してください。
[間隔]	Integration Server が POP3 または IMAP サーバに電子メールがあるかどうかをチェックする頻度 (秒単位)。
[メールのチェック後、常にログアウトする]	IMAP およびマルチスレッド処理のみで使用します。[**はい**] を選択した場合、Integration Server は読み取り専用スレッドにメールがあるかどうかをチェックした後、IMAP メールサーバに対する読み取り専用スレッドをログアウトするよう設定されます。IMAP サーバに対するメインの読み取り/書き込みスレッドは元のままです。[**いいえ**] を選択した場合、読み取り専用スレッドはすべて元のままになります。ログイン状態を維持できる接続数が IMAP サーバによって制限されている場合は [はい] を選択します。

8. [**セキュリティ**] で、以下の情報を指定します。

パラメータ	指定する値
[ユーザを指定してサービスを実行]	[**メッセージ内に認証が必要**] フィールドで [**はい**] を選択した場合は、[**ユーザを指定してサービスを実行**] フィールドは空白のままにします。Integration Server によって、ユーザ名およびパスワードが電子メールに記載されていると想定されるからです。[**メッセージ内に認証が必要**] フィールドで [**いいえ**] を選択した場合は、Integration Server 上でサービスを実行するユーザを入力する必要があります。
[メッセージ内に認証が必要]	[**はい**] を選択すると、電子メールの件名行に `$user` パラメータおよび `$pass` パラメータがあるかどうかが Integration Server によってチェックされます。ユーザ名は、Integration Server 上でサービスを実行するユーザになります。[**いいえ**] を選択した場合は、上記の [**ユーザを指定してサービスを実行**] フィールドでユーザを指定する必要があります。 [**いいえ**] を選択した場合、このフィールドの横に が表示されます。をクリックしてユーザを検索して選択します。内部ユーザでも外部ユーザでも使用できます。
[JSSE の使用]	JSSE ライブラリを使用して、TLS 1.1、TLS 1.2、または TLS 1.3 をサポートするポートを作成するには、[**はい**] を選択します。 Entrust IAIK ライブラリを使用して、セキュアでない TLS 1.0 のみをサポートするポートを作成するには、[**いいえ**] を選択します。

9. [**メッセージの処理**] で、次の情報を入力します。

パラメータ	指定する値
[グローバルサービス (オプション)]	Integration Server で実行されるサービス。このフィールドは、電子メールの件名行で指定したサービスよりも優先されます。
[デフォルトサービス (オプション)]	電子メールの件名行に有効なサービスの記載がなく、かつ、[グローバルサービス] フィールドが空白の場合に実行されるサービス。
[サービス出力を返信メールで送る]	サービスによって生成された出力を、電子メール添付の形式で送信元に返送するよう Integration Server に通知するには、[**はい**] をクリックします。送信元に報告しない場合は、[**いいえ**] をクリックします。元の電子メールに複数のファイルが添付されていた場合は、返信にも同数のファイルが添付されます。
[エラーを返信メールで送る]	電子メールの本文の部分を使用して、サービスの実行中に発生したエラーを送信元に報告するよう Integration Server に通知するには、[**はい**] をクリックします。送信元に報告しない場合は、[**いいえ**] をクリックします。 **注:** エラーが発生した場合に電子メール通知を送信する方法の詳細については、241 ページの「重大な問題に関するメッセージの電子メールアドレスへの送信」を参照してください。
[有効なメッセージを削除 (IMAP のみ)]	Integration Server による電子メールの正常受信後に、IMAP サーバから有効な電子メールを削除するには、[**はい**] をクリックします。この設定を使用すると電子メールが IMAP サーバに蓄積されなくなるため、ディスク容量の消費やパフォーマンスの低下を抑制することができます。Integration Server によって削除されるのは、常に、POP3 サーバ上の電子メールです。IMAP サーバ上に電子メールを保持しておく場合は、[**いいえ**] をクリックします。
[無効なメッセージを削除 (IMAP のみ)]	IMAP サーバ上の無効な電子メールを削除するには、[**はい**] をクリックします。サーバ上の無効な電子メールを削除したくない場合は、[**いいえ**] をクリックします。 無効な電子メールとは、呼び出し不可能なサービスを参照している電子メールをいいます。たとえば、参照されているサービスが存在しない場合、この電子メールはサーバによって削除されます。サービスは呼び出されたがそこでエラーが発生した場合、関連付けられている電子メールは有効であると見なされます。 この設定を使用すると無効な電子メールが IMAP サーバに蓄積されなくなるため、ディスク容量の消費やパフォーマンス

パラメータ	指定する値
	の低下を抑制することができます。Integration Server によって削除されるのは、常に、POP3 サーバ上の電子メールです。
[マルチスレッド処理 (IMAP のみ)]	このポートでマルチスレッド処理を使用するよう Integration Server に通知するには、[**はい**] をクリックします。この設定を使用すると、ポートで一度に複数の要求を処理できるため、ボトルネックを回避することができます。この機能が不要な場合は、[**いいえ**] をクリックします。
[スレッドの数 (マルチスレッド処理がオンになっている場合)]	このポートに対して使用するスレッドの数を Integration Server に通知します。デフォルトは 0 に設定されています。 **注:** 電子メールポートがマルチスレッド処理を実行できるように設定されている場合や電子メールポートで短期間に大量のメッセージを受信する場合は、メッセージを抽出して処理するために電子メールポートによってサーバスレッドプールが独占されます。この独占により、Integration Server のパフォーマンスが低下します。パフォーマンスの低下を回避するために、`watt.server.email.waitForServiceCompletion` プロパティを設定して、電子メールポートで受信されたメッセージの処理に使用されるスレッドの数を制限することができます。このプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。
[マルチパートメッセージの各パートに対してサービスを呼び出す]	マルチパートメッセージの各パートに対してサービスを呼び出すか、またはメッセージ全体に対して 1 回だけサービスを呼び出すかのいずれかを指定します。 [**いいえ**] を選択した場合、電子メールメッセージ全体が適切なコンテンツハンドラに渡された後、指定されているサービスに渡されて、サービスが実行されます。マルチパートメッセージ全体を送信する際には、コンテンツハンドラとサービスの両方またはいずれか一方が、メッセージの各パートに含まれているコンテンツタイプヘッダーの処理方法を認識できるように、メッセージの最初の部分に電子メールのヘッダーを含めてください。次の [**コンテンツハンドラにメッセージを受け渡すときに電子メールのヘッダを含める**] を参照してください。 [**はい**] を選択した場合、マルチパートメッセージのパートごとに処理されます。つまり、各パートがそれぞれコンテンツハンドラに渡された後、指定されているサービスに渡されます。[**はい**] を選択したときには、コンテンツハンドラとサービスの両方またはいずれか一方が、メッセージの各セクションに含まれているヘッダーの処理方法を認識済みであるため、通常は、メッセージの最初の部分に電子メールのヘッダーを

パラメータ	指定する値
	含める必要はありません。次の [**コンテンツハンドラにメッセージを受け渡すときに電子メールのヘッダを含める**] を参照してください。
[コンテンツハンドラにメッセージを受け渡すときに電子メールのヘッダを含める]	電子メールメッセージをコンテンツハンドラに渡す際に電子メールのヘッダーを含めるかどうかを指定します。電子メールのヘッダーは、一般的に電子メールメッセージの最初の部分にあります。マルチパートメッセージを単一メッセージとして処理する場合、[**はい**] を選択します。そうすることで、コンテンツハンドラとサービスの両方またはいずれか一方が、電子メールの本文を正しく処理できます。電子メールを異なるパートごとに個別に処理する場合、[**いいえ**] を選択します。単一パートの電子メールを処理する場合は、通常、電子メールのヘッダーを含める必要はありません。
[電子メールの本文にエンコードされたURL入力パラメータが含まれる]	電子メールメッセージに入力パラメータが見つかった場合の Integration Server での処理方法を選択します。この値を [**はい**] に設定すると、Integration Server によって `?one=1+two=2` のようなストリングはエンコードされた URL 入力パラメータと見なされます。その後、このストリングは IData オブジェクトにデコードされ、パイプラインに入れられて、サービスに渡されます。この値を [**いいえ**] に設定すると、Integration Server によってストリングはプレーンテキストとして扱われ、適切なコンテンツハンドラに渡されます。

10. [**変更内容の保存**] をクリックします。OAuth を設定した場合、Integration Server はアクセストークンを受信し、[**アクセストークンの有効期限**] を表示します。

> **注:**
> 電子メールポートを作成するか編集する場合、常にブラウザのキャッシュを消去する必要があります。これは、OAuth 認証タイプを使用している場合にのみ適用されます。

11. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

> **注:**
> ポートアクセス制限を設定する場合は、watt.net.email.validateHost サーバ設定プロパティを「`true`」に設定すると、Integration Server で確実に IP アクセス制限が実施されます。

ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

12. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

電子メールポートのセキュリティに関する考慮事項

ユーザ ID およびパスワードの受け渡しに電子メールを使用すると、セキュリティが公開される危険性があります。電子メールが POP3 または IMAP サーバに置かれている間に、ユーザ ID およびパスワードを使用した電子メールの情報が誰かにアクセスされてしまう可能性があります。電子メールサーバとの通信を安全なものにするために、Secure Sockets Layer (SSL) を介した明示的または暗黙的な転送レイヤセキュリティ (TLS) を使用するように Integration Server を設定できます。Integration Server は、クライアント認証を使用して転送レイヤセキュリティを実装し、ユーザ ID およびパスワードを使用して接続を認証します。

HTTP 診断ポートの追加

診断ポートは、要求を HTTP 経由で受信するために専用スレッドプールのスレッドを使用する特殊なポートです。診断ポートでは、専用のスレッドプールが使用されるので、Integration Server が無応答になったときにも Integration Server にアクセスできます。

Integration Server をインストールすると、診断ポートが自動的に 9999 に作成されます。その Integration Server の別のポートが 9999 で動作している場合、診断ポートは起動時に無効になります。

各 Integration Server にはそれぞれ 1 つの診断ポートしか設定できません。新しい診断ポートを追加する場合は、まず既存のポートを削除する必要があります。ポートの削除方法の詳細については、228 ページの「ポートの削除」を参照してください。

じマシン上で複数の Integration Server を実行している場合、インスタンスの作成プロセス中に、各サーバの診断ポート番号を指定しています。診断ポート番号が Integration Server インスタンス間で一意でない場合、最初にそのマシン上で開始された Integration Server では診断ポートが機能しますが、2 番目以降に開始した Integration Server では機能しません。同じマシン上で複数の Integration Server を実行する方法の詳細については、63 ページの「複数の Integration Server インスタンスの実行」を参照してください。

診断ポートの詳細については、993 ページの「Integration Server の診断 」を参照してください。

HTTP 診断ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**HTTP 診断**] を選択します。
5. [**サブミット**] をクリックします。
6. [**診断ポート設定の編集**] ページで、次の情報を入力します。

パラメータ	指定内容
[ポート]	診断ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **注:** watt.server.diagnostic.port サーバ設定は、このポート番号よりも優先します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバの診断ポート番号がそれぞれ固有であることを確認してください。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。デフォルトパッケージは WmRoot です。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。 **注:** このポートに関連付けられた [**パッケージ名**] は変更できません。診断ポートは常に WmRoot パッケージに関連付けられている必要があります。
[バインドアドレス(オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Integration Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 200 です。最大値は 65535 です。 **注:**

パラメータ	指定内容
	[バックログ] パラメータは無効化されたポートには適用されません。Integration Server は、無効化されたポートへ送信する要求を拒否します。
[キープアライブタイムアウト]	タイムアウト値 (ミリ秒単位) に指定された時間内にサーバがクライアントから要求を受信しなかった場合に接続を終了するタイミング、または、クライアントがサーバに終了要求を明示的に送信した場合に接続を終了するタイミング。
[スレッドプール]	リスナーが要求のディスパッチにこのプールを排他的に使用するかどうか。Integration Server の既存のスレッドプールはグローバルスレッドプールです。リソースの負荷が非常に高い場合は、グローバルスレッドプールを通じて要求を処理するために長い待ち時間が発生することがあります。ただし、プライベートスレッドプールオプションを有効化すれば、このポートで受信した要求が他のサーバ機能とスレッドを競合することを回避できます。 プライベートスレッドプールの設定を有効にするには、[**有効**] をクリックします。以下のデフォルト設定をそのまま使用するか、変更することができます。 [**スレッドプールの最小値**] は、プライベートスレッドプールのスレッドの最小数を示します。デフォルトは 1 です。 [**スレッドプールの最大値**] は、プライベートスレッドプールのスレッドの最大数を示します。デフォルトは 5 です。 [**スレッドの優先順位**] は、Java スレッドの優先順位を示します。デフォルトは 5 です。 **重要:** この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。 スレッドプール機能を使用しない場合は、[**無効**] をクリックします。 ポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数がサーバから報告されます。

7. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定する値
[クライアントの認証]	この HTTP ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。次のいずれかを選択します。

オプション	説明
[ユーザ名/パスワード]	Integration Server はクライアントに対してユーザ ID とパスワードを要求します。
[Digest]	Integration Server は、すべての要求を認証するために、パスワードダイジェストを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してダイジェストスキームで HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。 **注:** クライアント要求の認証用にパスワードダイジェストを使用するように設定されたポートは、ユーザが認証にパスワードダイジェストを許可するように設定されている場合に限り、そのユーザからの要求を処理します。ダイジェスト認証用にユーザを設定する方法の詳細については、104 ページの「ユーザアカウントの追加」を参照してください。
[ケルベロスチケットを要求する]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTP 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server は基本認証のユーザ名とパスワードを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。

パラメータ	指定する値	
	[ケルベロスチケットを必須にする]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTP 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server の認証は失敗します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。
[ケルベロスプロパティ (オプション)]	ケルベロスプロパティを使用して、ケルベロスチケットと共に受け取られるサービス要求の処理に使用するケルベロス関連の詳細を提供して、ケルベロス認証を有効にします。ケルベロス認証の設定の詳細については、552ページの「ケルベロス認証」を参照してください。	

	[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキストを指定します。
		
		以下の例で、**JAAS コンテキスト**は `KerberosClient` です。
		
		`KerberosClient {`
		`com.sun.security.auth.module.`
		`Krb5LoginModule required`
		`useKeyTab=true`
		`keyTab=alice.keytab;`
		`};`
		
		Integration Server で配布される is_jaas.cnf ファイルには、受信要求で使用できる IS_KERBEROS_INBOUND という名前の JAAS コンテキストが含まれます。

	[プリンシパル]	ケルベロス認証で使用されるプリンシパルの名前を指定します。
	[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワードを指定します。プリンシパルと

パラメータ	指定する値	
		そのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。 JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
	[プリンシパルパスワードの再入力]	プリンシパルパスワードを再入力します。
	[サービスプリンシパル名形式]	[**ユーザ名**] が表示されます。KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義されている名前付きユーザとしてサービスのプリンシパル名が表されることを示します。
	[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスで使用するプリンシパルの名前を指定します。**サービスプリンシパル名**は、以下の形式で指定します。 *principal-name.instance-name@realm-name*

8. [**変更内容の保存**] をクリックします。

9. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

   ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。519 ページの「リソースへのアクセスをポートごとに制御」

10. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

HTTPS 診断ポートの追加

診断ポートは、要求を HTTP/S 経由で受信するために専用スレッドプールのスレッドを使用する特殊なポートです。診断ポートでは、専用のスレッドプールが使用されるので、Integration Server が無応答になったときにも Integration Server にアクセスできます。

各 Integration Server にはそれぞれ 1 つの診断ポートしか設定できません。新しい診断ポートを追加する場合は、まず既存のポートを削除する必要があります。ポートの削除方法の詳細については、228 ページの「ポートの削除」を参照してください。

じマシン上で複数の Integration Server を実行している場合、インスタンスの作成プロセス中に、各サーバの診断ポート番号を指定しています。診断ポート番号が Integration Server インスタンス間で一意でない場合、最初にそのマシン上で開始された Integration Server では診断ポートが機能しますが、2 番目以降に開始した Integration Server では機能しません。同じマシン上で複数の Integration Server を実行する方法の詳細については、63 ページの「複数の Integration Server インスタンスの実行」を参照してください。

HTTPS 診断ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**HTTPS 診断**] を選択します。
5. [**サブミット**] をクリックします。
6. [**診断ポート設定の編集**] ページの [**診断 HTTPS リスナー設定**] で、次の情報を入力します。

パラメータ	指定する値
[有効]	この HTTPS 診断ポートを有効にするか ([**はい**]) 無効にするか ([**いいえ**]) を選択します。
[ポート]	診断ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **注:** watt.server.diagnostic.port サーバ設定は、このポート番号よりも優先します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバの診断ポート番号がそれぞれ固有であることを確認してください。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。

パラメータ	指定する値
[パッケージ名]	このポートに関連付けるパッケージ。デフォルトパッケージは WmRoot です。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。 **注:** このポートに関連付けられた [**パッケージ名**] は変更できません。診断ポートは常に WmRoot パッケージに関連付けられている必要があります。
[バインドアドレス(オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Integration Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 200 です。最大値は 65535 です。 **注:** [バックログ] パラメータは無効化されたポートには適用されません。Integration Server は、無効化されたポートへ送信する要求を拒否します。
[キープアライブタイムアウト]	タイムアウト値 (ミリ秒単位) に指定された時間内にサーバがクライアントから要求を受信しなかった場合に接続を終了するタイミング、または、クライアントがサーバに終了要求を明示的に送信した場合に接続を終了するタイミング。
[スレッドプール]	リスナーが要求のディスパッチにこのプールを排他的に使用するかどうか。Integration Server の既存のスレッドプールはグローバルスレッドプールです。リソースの負荷が非常に高い場合は、グローバルスレッドプールを通じて要求を処理するために長い待ち時間が発生することがあります。ただし、プライベートスレッドプールオプションを有効化すれば、このポートで受信した要求が他のサーバ機能とスレッドを競合することを回避できます。プライベートスレッドプールの設定を有効にする場合は、[**有効**] をクリックします。以下のデフォルト設定をそのまま使用するか、変更することができます。

パラメータ	指定する値
	[**スレッドプールの最小値**] は、プライベートスレッドプールのスレッドの最小数を示します。デフォルトは 1 です。 [**スレッドプールの最大値**] は、プライベートスレッドプールのスレッドの最大数を示します。デフォルトは 5 です。 [**スレッドの優先順位**] は、Java スレッドの優先順位を示します。デフォルトは 5 です。 **重要:** この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。 スレッドプール機能を使用しない場合は、[**無効**] をクリックします。 ポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数がサーバから報告されます。

7. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定する値
[**クライアントの認証**]	この HTTPS ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。詳細については、549 ページの「クライアントの認証」を参照してください。 次のいずれかを選択します。 **オプション** / **説明** [**ユーザ名/パスワード**] Integration Server はクライアントに対してユーザ ID とパスワードを要求します。 [**Digest**] Integration Server は、すべての要求の認証に対してパスワードダイジェストを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してダイジェストスキームで HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。

パラメータ	指定する値
	注: クライアント要求の認証用にパスワードダイジェストを使用するように設定されたポートは、ユーザが認証にパスワードダイジェストを許可するように設定されている場合に限り、そのユーザからの要求を処理します。ダイジェスト認証用にユーザを設定する方法の詳細については、104 ページの「ユーザアカウントの追加」を参照してください。
[クライアント認証を要求する]	Integration Server はすべての要求に対してクライアント認証を要求します。クライアントから認証が提示されなかった場合、サーバはクライアントにユーザ ID とパスワードを要求します。クライアントから認証が提供された場合は、次の処理が実行されます。 ■ Integration Server は、認証がファイル上のクライアント認証と正確に一致し、信用のある認証局によって署名されているかどうかをチェックします。上記のチェックに該当する場合、クライアントは Integration Server 内で認証がマッピングされているユーザとしてログインします。該当しない場合、セントラルユーザ管理が設定されていない限り、クライアント要求は失敗します。 ■ セントラルユーザ管理が設定されている場合、Integration Server は、セントラルユーザデータベース内で認証がユーザにマッピングされているかどうかをチェックします。マッピングされている場合、クライアントはそのユーザとしてサーバにログインします。マッピングされていない場合、クライアント要求は失敗します。
[クライアント認証を必須にする]	Integration Server はすべての要求に対してクライアント認証を必須にします。サーバは、クライアントが常に認

パラメータ	指定する値	
		証を提示する必要があることを除いて、[**クライアント認証を要求する**] を指定した場合と同様に動作します。
	[ID プロバイダを使用]	Integration Server は OpenID プロバイダを使用して要求を認証します。Integration Server はこのポートに送信されるすべての要求を [**ID プロバイダ**]. に指定された OpenID プロバイダにリダイレクトします。
	[ケルベロスチケットを要求する]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTP 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server は基本認証のユーザ名とパスワードを使用します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。
	[ケルベロスチケットを必須にする]	Integration Server は、ネゴシエーション認証スキームを使用して、HTTP 認証ヘッダーのケルベロスチケットを探します。チケットが見つからない場合は、Integration Server の認証は失敗します。クライアントが認証情報を提供しない場合、Integration Server は認証情報を要求するクライアントに対してネゴシエーションスキームと共に HTTP WWW-Authenticate ヘッダーを返します。クライアントが必要な認証情報を提供する場合、Integration Server は要求を検証および検査します。
[ケルベロスプロパティ (オプション)]	ケルベロスプロパティを使用して、ケルベロスチケットと共に受け取られるサービス要求の処理に使用するケルベロス関連の詳細を提供して、ケルベロス認証を有効にします。ケルベロス認証の設定の詳細については、552 ページの「ケルベロス認証」を参照してください。	

パラメータ	指定する値	

	[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキストを指定します。
		
		以下の例で、**JAAS コンテキスト**は `KerberosClient` です。
		`KerberosClient {`
		`com.sun.security.auth.module.`
		`Krb5LoginModule required`
		`useKeyTab=true`
		`keyTab=alice.keytab;`
		`};`
		
		Integration Server で配布される is_jaas.cnf ファイルには、受信要求で使用できる IS_KERBEROS_INBOUND という名前の JAAS コンテキストが含まれます。

	[プリンシパル]	ケルベロス認証で使用されるプリンシパルの名前を指定します。
	[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワードを指定します。プリンシパルとそのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。 JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
	[プリンシパルパスワードの再入力]	プリンシパルパスワードを再入力します。
	[サービスプリンシパル名形式]	[**ユーザ名**] が表示されます。KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義されている名前付きユーザとしてサービスのプリンシパル名が表されることを示します。

パラメータ	指定する値
[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスで使用するプリンシパルの名前を指定します。**サービスプリンシパル名**は、以下の形式で指定します。 *principal-name.instance-name@realm-name*
[JSSE の使用]	このポートで TLS 1.1、TLS 1.2、または TLS 1.3 をサポートする必要がある場合は、[**はい**] をクリックして、JSSE (Java Secure Socket Extension) ソケットファクトリを使用するポートを作成します。デフォルトは [**はい**] です。 この値を [**いいえ**] に設定すると、ポートはセキュアではない TLS 1.0 のみをサポートします。 **注:** JSSE を使用する Integration Server ポートで使用される暗号化スイートを制御し、受信要求を処理するには、watt.net.jsse.server.enabledCipherSuiteList を設定します。詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

8. [**リスナー固有のクレデンシャル (オプション)**] で、次の情報を入力します。

注:
[認証] ページで指定した認証とは異なる認証セットを使用する場合にのみ、この設定を使用します。

パラメータ	指定する値
[キーストアエイリアス]	オプション。Integration Server キーストアのユーザ指定のテキスト識別子。 エイリアスは秘密鍵および関連する認証のリポジトリを指し示します。それぞれのリスナーで 1 つのキーストアを指し示しますが、同じキーストア内に複数のキーおよび認証が存在し、複数のリスナーで同じキーストアエイリアスを使用することもできます。 詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
[キーエイリアス]	オプション。上記のキーストアエイリアスによって指定されるキーストアに保存されている必要がある、秘密鍵のエイリアス。
[トラストストアエイリアス]	オプション。トラストストアのエイリアス。トラストストアには、キーエイリアスに関連付けられている Integration Server

パラメータ	指定する値
	認証に署名した認証局の信用のあるルート認証が格納されている必要があります。また、トラストストアには、Integration Server が信用関係の妥当性検査を行うために使用する認証局の認証のリストも格納されます。

9. [**変更内容の保存**] をクリックします。

10. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

    ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

11. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

HTTP/HTTPS ポートの一時停止

デフォルトでは、Integration Server はポート接続要求を受信すると直ちに要求を受け入れます。ポートの一時停止を設定すると、ポートで接続を受け入れたり要求をこれ以上ディスパッチしたりしないようにできます。

注:
一時停止中の受信待機ポートに対して要求が送られたときにバックログキューが無効になっていた場合、リスナーはこれ以上の接続の受け入れや要求のディスパッチを行いません。ただし、バックログキューが有効化されていて、満杯でなければ、接続はキューイングされます。一時停止中のポートを削除または無効化すると、キューイングされていた接続が解放されます。

HTTP または HTTPS ポートを一時停止するには、以下の手順に従います。

HTTP ポートまたは HTTPS ポートを一時停止するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. [**ポートの一覧**] テーブルで、一時停止するポートの [**拡張**] 列の [**編集**] をクリックします。

4. [**リスナーコントロール**] の [**一時停止**] チェックボックスをオンにします。

5. [**適用**] をクリックして変更内容を保存します。

6. [**ポートに戻る**] をクリックします。

HTTP/HTTPS ポートの再開

一時停止したポートを再開できます。

HTTP ポートまたは HTTPS ポートを再開するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの一覧**] テーブルで、再開するポートの [**拡張**] 列の [**編集**] をクリックします。
4. [リスナーコントロール] の [**再開**] チェックボックスをオンにします。
5. [**適用**] をクリックして変更内容を保存します。
6. [**ポートに戻る**] をクリックします。

HTTPS 要求のテスト

指定したポートでサーバが HTTPS 要求を受信待機しているかどうかをテストするには、ブラウザを起動して「`https://localhost:port`」と入力します。

- ポートが正常に動作していれば、Integration Server Administrator のログオンページが表示されます。
- Integration Server Administrator が表示されない場合は、マシン上で実行しているサービスが同じポートで受信待機していないかどうかを確認します。

FTP/FTPS ポート範囲の使用

Integration Server は、任意のフリーポート上で FTP/FTPS クライアントデータ接続を受信待機する FTP/FTPS リスナーを提供します。FTPS の場合、このような方法でポートを使用する際には、ファイアウォール上ですべてのポートを開いておく必要があります。これは、ファイアウォールの管理者にとっては歓迎できない状況です。

受信転送モード (PASV) を使用するクライアントのデータ接続に使用する FTP/FTPS リスナーのポート番号の範囲を指定できます。Integration Server では、以下の設定パラメータを使用して、ポート範囲の上限および下限を指定できます。

- watt.net.ftpPassivePort.min
- watt.net.ftpPassivePort.max

FTP/FTPS ポート範囲の指定

FTP および FTPS ポートのポート範囲を設定する場合は、以下の操作上の考慮事項に留意してください。

- watt.net.ftpPassivePort.min および watt.net.ftpPassivePort.max パラメータが存在しない、または未定義の場合、FTP/FTPS リスナーは、任意のフリーポート上で受信待機するという従来の動作を維持します。
- watt.net.ftpPassivePort.min に指定された値が 1 未満のときには、1 というデフォルト値が使用されます。watt.net.ftpPassivePort.max に指定された値が 65534 を超えるときには、65534 というデフォルト値が使用されます。これら両方の条件が同時に存在する場合、FTP/FTPS リスナーは、任意のフリーポート上で受信待機するという従来の動作を維持します。
- 指定された値が予想された範囲内にない場合は、コマンドチャネルから FTP/FTPS クライアントにエラーメッセージが戻されます。たとえば、一方のプロパティが未定義の場合や、watt.net.ftpPassivePort.min の値が watt.net.ftpPassivePort.max の値よりも大きい場合、または一方のプロパティの数値が無効な場合などがこれにあたります。
- また、指定されたポート範囲のすべてのポートが使用中の場合も、エラーメッセージが戻されます。
- 個々のエラーメッセージの詳細については、serverYYYYMMDD.log ファイルに記録されます。
- ポート範囲のプロパティはいつでも Integration Server Administrator で変更できます。

FTP および FTPS ポートリスナーのポート範囲を指定するには、以下の手順に従います。

FTP および FTPS リスナーのポート範囲を指定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [拡張設定] に移動します。
3. 次のサーバ設定パラメータを設定します。

拡張設定項目	入力する値
`watt.net.ftpPassivePort.min`	*Minimum_Port_Number*
`watt.net.ftpPassivePort.max`	*Maximum_Port_Number*

Minimum_Port_Number および *Maximum_Port_Number* の値は、1 から 65534 の範囲のポート番号です。これらのプロパティを使用してポート範囲を指定した場合、指定された最小値と最大値の範囲内のポートのみが、受信 FTP/FTPS クライアントデータ接続の受信待機ポートとして使用されます。最小値と最大値の両方を指定する必要があります。

プライマリポートについて

プライマリポートは、ユーザが Integration Server の受信待機用メインポートとして指定する HTTP ポートまたは HTTPS ポートです。サーバはどのような特別な目的であってもプライマリポートを予約することはありません。ただし、Integration Server ではプライマリポートの削除は許可されないので、少なくとも 1 つのポートが常に使用可能であることが保証されます。

プライマリポートを指定しない場合は、起動時に Integration Server によって 1 つ作成されます。デフォルトの Integration Server 設定では、ポート番号 5555 の HTTP ポートがプライマリポートとして指定されています。同じマシン上で複数の Integration Server を作成した場合、インスタンスの作成プロセス中に、各サーバの固有のプライマリポート番号を指定しています。

プライマリポート番号は、クライアントからサーバプロパティ watt.server.port について問い合わせがあった場合にクライアントへ返されるポート番号でもあります。

注:
Broker の設定後に Integration Server のプライマリポート番号を変更した場合、Integration Server の Broker クライアントと Integration Server の設定との同期が失われることがあります。プライマリポートを変更した後、Broker クライアントを Integration Server の新しいポート設定に同期させる必要があります。詳細については、285 ページの「Integration Server のプライマリポートが変更された場合の Broker クライアントの同期」を参照してください。

プライマリポートの変更

ポートをプライマリポートとして指定する場合は、以下の点に留意してください。

- ポートは HTTP ポートまたは HTTPS ポートである必要があります。
- ポートは WmRoot パッケージに関連付ける必要があります。
- ポートは有効である必要があります。
- ポートは標準ポートである必要があります。つまり、プライマリポートを診断ポートにすることはできません。

プライマリポートを変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**プライマリポートの変更**] をクリックします。
4. ページの [**新規プライマリポートの選択**] 領域で、プライマリポートにするポートを [**プライマリポート**] リストから選択します。

   Integration Server Administrator には、プライマリポートになる条件を満たすポート (WmRoot パッケージに関連付けられた、有効な標準 HTTP ポートまたは HTTPS ポート) のみがリストされます。

5. [**更新**] をクリックします。

ポートの削除

ポートが不要になったら、役割を削除できます。

重要:
Integration Server に定義済みのプライマリポートは削除できません。

ポートを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの一覧**] でポートを見つけ、[**削除**] 列で ✕ アイコンをクリックします。アクションの確認を求めるダイアログボックスが表示されます。[**OK**] をクリックしてポートの削除を確認します。

ポートの編集

ポートを追加した後でポート設定を編集できます。ポートは設定を編集する前に無効にする必要があります。

注:
ポートの作成後は、ポートエイリアスを編集できません。

ポートを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. 編集するポートを見つけて、そのポート番号をクリックします。
4. [<ポートタイプ> **ポート設定の編集**] をクリックします。
5. ポートの情報を更新します。
6. [**変更内容の保存**] をクリックします。

ポートの有効化/無効化について

サーバの 1 個のポートに一時的に要求が承認されないようにするには、そのポートを無効にします。このアクションによって、要求は受信されてもサーバまで到達しなくなります。ポートが

無効になっているときにクライアントからポートへ要求が発行されると、クライアントにエラーメッセージが返されます。ポートは後で有効化できますが、サーバをシャットダウンして再起動しても、管理者が有効にするまでポートは無効のままです。このようにポートを無効化すると、Integration Server の本番稼動が始まってからは開発者がサーバにアクセスできないようにする場合に便利です。

ポートを有効または無効にするもう 1 つの方法は、ポートに関連付けられたパッケージを有効または無効にする方法です。パッケージを特定のポートに関連付けることにより、パッケージを複製しても同じ番号のポートを新しいサーバで引き続き使用できます。パッケージがポートに関連付けられている場合、パッケージを有効にするとポートも有効になり、パッケージを無効にするとポートも無効になります。

重要:
少なくともプライマリポートだけは有効のままにしておく必要があります。

ポートの無効化

ポートを無効にするには、以下の手順に従います。

ポートを無効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの一覧**] でポートを見つけ、[**有効**] 列で ✓ アイコンをクリックしてポートを無効にします。アクションの確認を求めるダイアログボックスが表示されます。[**OK**] をクリックしてポートの無効化を確認します。

   ✓ アイコンが [**いいえ**] に置き換えられ、ポートが無効になったことが示されます。

HTTP/HTTPS ポートを無効にした場合、Integration Server では新しい要求の受け入れを即座に停止します。ただし、実行中のすべての要求は完了できます。Integration Server では、実行中の要求の応答ヘッダーに `Connection: close` ヘッダーフィールドを追加します。これは、Integration Server では実行中の要求の完了後に接続を閉じることを示します。

HTTP/HTTPS 1.1 要求の場合、Integration Server では常に応答ヘッダーに `Connection: close` ヘッダーフィールドを追加します。HTTP/HTTPS 1.0 の場合、クライアントの要求に `keep-alive` ヘッダーフィールドが含まれる場合にのみ、Integration Server では `Connection: close` ヘッダーフィールドを追加します。

ポートの有効化

ポートを有効にするには、以下の手順に従います。

ポートを有効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. [**ポートの一覧**] で有効にするポートを見つけ、その [**有効**] 列にある [**いいえ**] をクリックしてポートを有効にします。アクションの確認を求めるダイアログボックスが表示されます。[**OK**] をクリックしてポートを有効にします。

   [**いいえ**] が ✓ アイコンに置き換えられ、ポートが有効になったことが示されます。

ポートによるクライアント認証の処理の設定

ここでは、Integration Server Administrator を使用してポートによるクライアント認証の処理方法を表示または変更する方法について説明します。クライアント認証の詳細については、549 ページの「クライアントの認証」を参照してください。

ポートがクライアント認証を処理する方法を表示または変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. クライアント認証の設定を表示、変更または無効にする (無効になっていない場合) ポートを選択します。

   注:
   ポートを無効にするには、[**有効**] 列にある ✓ アイコンをクリックします。✓ アイコンが [**いいえ**] に置き換えられ、ポートが無効になったことが示されます。

4. ポート番号をクリックします。

5. [**HTTPS ポート設定の編集**] または [**FTPS ポート設定の編集**] をクリックし、フィールド内の情報を必要に応じて更新します。フィールドの説明については、180 ページの「HTTPS ポートの追加」または 194 ページの「FTPS ポートの追加」を参照してください。

6. [**変更内容の保存**] をクリックします。

7. [**有効**] 列の [**いいえ**] をクリックして、ポートを有効にします。

セキュリティプロバイダの追加

キーストア内の秘密鍵および認証チェーンを使用するリスナーを指定して HTTPS または FTPS ポートを追加するときに、キーストアが非標準のセキュリティプロバイダによって管理されていると、Integration Server Administrator にセキュリティプロバイダを追加することが必要になる場合があります。

HTTPS または FTPS のポート情報ページでキーストア情報を指定する際、非標準のセキュリティプロバイダは、[キーストアタイプ] パラメータのドロップダウンリストに表示されない場合があります。使用するセキュリティプロバイダがリストに表示されない場合、[セキュリティプロバイダの追加] リンクを使用して、セキュリティプロバイダを追加します。

セキュリティプロバイダを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [キーストア] に移動します。

3. [**セキュリティプロバイダの追加**] をクリックします。

4. ページの [**セキュリティプロバイダの追加**] 領域の [**セキュリティプロバイダクラス**] フィールドに、追加のキーストアおよびトラストストアのファイルタイプに使用するセキュリティプロバイダの Java クラス名を入力します。たとえば、nCipher のセキュリティプロバイダ名は、`com.ncipher.provider.km.nCipherKM` です。

   注:
   対応する jar ファイルがクラスパスに存在していることを確認します。

   HSM でサポートされているキーストアタイプが Integration Server でデフォルトでサポートされているタイプではない場合、watt.security.keyStore.supportedTypes プロパティまたは watt.security.trustStore.supportedTypes プロパティをそれぞれ変更して、キーストアまたはトラストストアの新しいキーストアタイプを追加します。

5. [**プロバイダの追加**] をクリックします。

   Integration Server によって、使用可能なセキュリティプロバイダのリストにセキュリティプロバイダが追加されます。新たに追加したセキュリティプロバイダでサポートされているキーストアタイプが Integration Server でサポートされているデフォルトのキーストアタイプ (JKS、PKCS12) のいずれかで、HTTPS または FTPS ポートの [キーストアタイプ] リストでそのキーストアタイプを選択している場合、対応するプロバイダがそのキーストアタイプの [プロバイダ] リストに表示されるようになります。

   キーストアタイプがサポートされていない場合は、watt.security.keyStore.supportedTypes および watt.security.trustStore.supportedTypes プロパティを変更して、キーストアおよびトラストストアの新しいキーストアタイプを追加します。

ポートごとに JSSE 用のプロトコルを無効にする

ポートごとに JSSE 用のプロトコルを無効にすることができます。これは、HTTPS ポートと FTPS ポートにのみ適用されます。ポートごとに無効にされたプロトコルは、watt.net.jsse.server.disabledProtocols に指定されたプロトコルよりも優先されます。後者の設定はサーバ内で受信 JSSE 接続用のプロトコルの無効化を指定します。ポートでサポートされていないプロトコルの方が、watt.net.jsse.server.disabledProtocols パラメータによってサーバ全体で無効にされたプロトコルより多少制約が多いことがあります。

ポートごとに JSSE 用のプロトコルを無効にするには

1. 52 ページの「Integration Server のシャットダウン」の説明に従って、Integration Server をシャットダウンします。

2. 次のファイルをテキストエディタで開きます。

   Integration Server_directory /instances/*instanceName*/packages/*packageName*/listeners.cnf

   instanceName は Integration Server インスタンスの名前、*packageName* はポートに関連付けられているパッケージの名前です。

3. listeners.cnf ファイルで、無効にするプロトコルを指定する HTTPS および FTPS ポートのレコードを検索します。

   たとえば、次のように入力します。

   - HTTPS ポート 5333 に変更する場合は、ポートレコードは次のように始まります。

     ```
     <record name="HTTPSListener@5333" javaclass="com.wm.util.Values">
     ```

   - FTPS ポート 4602 に変更する場合は、ポートレコードは次のように始まります。

     ```
     <record name="FTPSListener@4602" javaclass="com.wm.util.Values">
     ```

4. ポートレコードの `<value name="useJSSE">true</value>` エントリの後に、次のエントリを追加します。

   ```
   <value name="jsseEnabledProtocols">SSLprotocols</value>
   ```

   SSLprotocols は、ポートがサポートしない SSL プロトコルバージョンのカンマ区切りのリストです。

   たとえば、ポートの TLS 1.2 および TLS 1.3 バージョンを有効にし (TLS 1.3 が Integration Server でサポートされている場合)、他のすべてのプロトコルを無効にする場合、次の行を追加します。

   ```
   <value name="jsseDisabledProtocols">SSLv2Hello,SSLv3,TLSv1,TLSv1.1</value>
   ```

   注:
   ネゴシエーション中のプロトコルダウングレードを避けるには、*SSLprotocols* を設定し、TLSv1.2 以降の 1 つまたは 2 つのプロトコルを除く、すべてのプロトコルバージョンを無効にします。

5. 変更を保存してテキストエディタを閉じます。

6. Integration Server を再起動します。

   注:

listeners.cnf ファイルで指定されるポートレコードの `jsseDisabledProtocols` 値は、watt.net.jsse.server.disabledProtocols サーバ設定パラメータによって設定される値を上書きします。

デフォルトの Info ログレベルで、Integration Server は、受信/送信ポートの起動時に有効なプロトコルを示すメッセージを書き込みます。ログ機能 0006 サーバ SSL インタフェースがデバッグログレベルに設定されている場合、Integration Server は受信/送信ポートに使用されるサポートされて有効なプロトコルに関するメッセージをサーバログに書き込みます (Integration Server は送信 SSL 接続の確立の一部として送信ポートを作成します)。トレースログレベルでは、Integration Server は有効な暗号スイートに関するメッセージを書き込みます。これらのサーバログメッセージを使用して、任意の JSSE ポートで有効になっているプロトコルを確認できます。

HTTP/S ポートをステートレスに指定する

Integration Server では、Integration Server の HTTP ポートと HTTPS ポートをステートレスにできるようになりました。ステートレスポートでは、ポートで受信された要求のセッションまたはセッション ID の提供および保持を行いません。これは、負荷分散機能を持つ複数の非クラスタ Integration Server を使用する場合に役立つことがあります (この設定はステートレスクラスタと呼ばれることがあります)。

サーバ設定パラメータ watt.server.portStateless を使用して、ステートレスである Integration Server のポートのポート番号のカンマ区切りリストを指定します。次に例を示します。

```
watt.server.portStateless=8443

watt.server.portStateless=8443,8081
```

プライマリポートおよび診断ポートを除く、Integration Server の任意の HTTP ポートまたは HTTPS ポートをステートレスに設定できます。また、WebSockets 用、または Integration Server の管理に使用するポートはステートレスに設定しないでください。プライマリポート、診断ポート、HTTP ポートまたは HTTPS ポートではないポート、または watt.server.portStateless の値に存在しないポートを指定した場合、Integration Server では、ポートをステートレスにできずスキップすることを示す警告メッセージをログに記録します。

11 サーバログの設定

- サーバログの概要 236
- サーバログに記録する情報の量と種類の指定 237
- サーバログエントリをキューに格納するかどうかの指定 239
- サーバログのデフォルト場所の変更 239
- サーバログをサイズに基づいて循環するように設定する 240
- Integration Server によって保持保持されるサーバログファイル数の制限 241
- 重大な問題に関するメッセージの電子メールアドレスへの送信 241
- ログエントリに対する追加処理の実行 243
- サーバログの表示 243
- ログ表示の変更 244
- セッションのログレベルとサーバログの場所の変更 247
- グローバリゼーションとサーバログ 248

サーバログの概要

Integration Server サーバログには、Integration Server サブシステムの起動、Integration Server または他の webMethods 製品に含まれるパッケージのロードなど、Integration Server で行われた処理および発生したエラーに関する情報が記録されます。エントリは、Integration Server の重要なサブシステム (機能) によって、サーバログに書き込まれます。たとえば、Integration Server パッケージ機能はパッケージをロードおよびアンロードするとき、Integration Server フローマネージャ機能はフローサービスを処理するとき、Integration Server の HTTP ポートは要求数を受信したときにサーバログエントリを書き込みます。また、Trading Networks Server またはアダプタなど、Integration Server にインストールされた各製品の機能もサーバログエントリを書き込みます。次に、サンプルサーバログの一部を示します。

```
2021-01-28 15:34:40 EST [ISS.0025.0006I] (tid=82) License Manager started
2021-01-28 15:34:41 EST [ISS.0025.0049I] (tid=82) The JCE Unlimited Strength
Jurisdiction Policy File was found }
2021-01-28 15:34:41 EST [ISS.0025.0041I] (tid=82) FIPS mode not initialized
2021-01-28 15:34:43 EST [ISS.0096.0002W] (tid=82) JDBCConnectionManager: creating
Embedded Pool Alias for Xref.
2021-01-28 15:34:43 EST [ISS.0096.0002W] (tid=82) JDBCConnectionManager: creating
Embedded Pool Alias for ISInternal.
2021-01-28 15:34:43 EST [ISS.0096.0003I] (tid=82) JDBCConnectionManager: creating
database object for embedded database: IS_USER_TASKS.LAST_UPDATED 2
021-01-28 15:34:43 EST [ISS.0025.0024I] (tid=82) JDBC Connection Manager started
2021-01-28 15:34:43 EST [ISS.0095.0026W] (tid=82) The AuditConfig.xml file was not
found and is being created.The audit logging system is using default settings.
2021-01-28 15:34:43 EST [ISS.0095.0025W] (tid=82) The auditing.cnf file was not found
 and is being created.2021-01-28 15:34:43 EST [ISS.0095.0025W] (tid=82) The
transient.sql.errors.xml file was not found and is being created.
2021-01-28 15:34:44 EST [ISS.0095.0013I] (tid=82) Audit Logging initialized.
2021-01-28 15:34:44 EST [ISS.0017.0080I] (tid=279) Migrated the SoftwareAG-IS-WMN.xml
 cache manager configuration to the Ehcache 2.8.x format.
2021-01-28 15:34:45 EST [ISS.0045.0096I] (tid=82) Startup action for the IS Statistics
 Data Collector is underway.
```

デフォルトでは、すべての機能がサーバログに書き込みを 行います。また、機能が書き込むログエントリは、重大、エラー、警告、情報およびデバッグのいずれかの状況に関連しています。ログへの書き込みは、選択した機能に限定したり、 機能から記録されるデータ量を増やしたり減らしたりすることもできます。このような柔軟性があるため、トラブルシューティングを行う場合に役立ちます。 たとえば、サーバログに記録される内容の詳細レベルを 一時的に上げると、Integration Server のエラーまたはパフォーマンス問題の原因特定に役立つ場合が あります。このレベルは、問題が解決した時点で元のレベルに 戻します。特定の Integration Server セッションについて、 サーバログに記録する情報の量を 変更することもできます。

デフォルトでは、Integration Server は機能によってメモリに書き込まれたログエントリをキューに格納し、バックグラウンドスレッドを使用してログエントリをサーバログに書き込みます。このプロパティを変更すると、機能はサーバログに直接書き込むようになります。バックグラウンドスレッドを使用する場合は Integration Server のパフォーマンスが向上しますが、直接書き込む場合はエントリがサーバログに表示されるまでの時間が短縮されます。

Integration Server は、常にサーバログエントリをフラットファイルに書き込みます。サーバログをデータベースに保存することはできません。デフォルトでは、Integration Server は、現在の日付のサーバログエントリを server.log ファイルに書き込みます (当日の定義は午前零時から午前零時まで)。ただし、日単位ではなくサイズ単位で server.log ファイルを循環するように

Integration Server を設定できます。詳細については、240 ページの「サーバログをサイズに基づいて循環するように設定する」を参照してください。

日単位またはサイズ単位で server log ファイルが循環される場合、Integration Server は現在の server.log ファイルの名前をアーカイブファイル名が付くように変更して、新しい server.log ファイルを開始します。アーカイブファイルの名前には、最初にログエントリが書き込まれた日付 (*yyyymmdd*) が追加されます。

デフォルトでは、ログファイルおよびアーカイブファイルは、すべて *Integration Server_directory* *instances*\instance_name\logs ディレクトリに保存されます。サーバログファイルの場所の変更の詳細については、239 ページの「サーバログのデフォルト場所の変更」を参照してください。

Integration Server は作成した各サーバログファイルを保持します。ただし、リソースを維持するため、Integration Server によって保持されるサーバログファイル数の制限が必要な場合があります。Integration Server によって保持されるサーバログファイル数の自動的な制限の詳細については、241 ページの「Integration Server によって保持保持されるサーバログファイル数の制限 」を参照してください。

特定のセッションについて、サーバログファイルの場所を変更することもできます。サーバログファイルの場所またはセッションのログレベルの上書きの詳細については、247 ページの「セッションのログレベルとサーバログの場所の変更」を参照してください。

サーバログに記録する情報の量と種類の指定

個別の製品や、製品内の特定の機能に対してログレベルを指定できます。機能は親の製品からログレベルを継承し、さらに製品はデフォルトノードからログレベルを継承するので、デフォルトノードのログレベルを設定することにより、大部分の製品と機能で使用するデフォルトのログレベルにすることができます。次に、サーバログ情報をもっと詳しく、または、もっと簡略して生成したい特定の製品または機能のログレベルを変更できます。

デフォルトでは、すべての製品がデフォルトノードのログレベルを継承します。継承した値は灰色のテキストで表示されます。製品または機能のログレベルを明示的に変更した場合、そのレベルはデフォルトノードのレベルよりも優先されます。Integration Server Administrator では、明示的に設定されたログレベルは**太字**テキストで表示されます。

サーバログに記録する情報の量と種類を指定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**ログ**] > [ログ設定] に移動します。
3. [**ロガーリスト**] で [**サーバロガー**] をクリックします。
4. [**サーバロガーの編集**] をクリックします。

   [**サーバロガーの設定**] 領域に、Integration Server にインストールされている Integration Server と製品およびそれぞれの機能が表示されます。機能および現在のログレベルを確認するには、⊞ をクリックして表示を展開します。

5. 次の操作を必要なだけ実行します。

ログレベル	アクション
デフォルトノード	デフォルトノードの [**ログレベル**] リストで、使用するレベルをクリックします。Integration Server Administrator によって、デフォルトノードから継承しているすべての製品および機能が新しいレベルにリセットされます。
特定の製品	製品ノードの [**ログレベル**] リストで、使用するレベルをクリックします。Integration Server Administrator によって、製品から値を継承しているすべての機能が新しいレベルにリセットされます。 特定の製品のレベルを変更し、その後、製品で再度デフォルトのノードからログレベルを継承することが必要になった場合は、製品のログレベルをデフォルトノードのログレベルにリセットします。
特定の機能	機能の [**ログレベル**] リストで、使用するレベルをクリックします。 特定の機能のレベルを変更し、その後、機能で再度製品からログレベルを継承することが必要になった場合は、機能のログレベルを製品のログレベルにリセットします。

ログレベルの詳細については、238 ページの「ログレベル」を参照してください。

重要:
記録する情報が増えると、システムのリソース消費も多くなります。

6. [**変更内容の保存**] をクリックします。

ログレベル

指定できるログレベルを次に示します。各ログレベルには、そのレベルのメッセージに加えて、それより上のレベルのメッセージもすべて含まれます (たとえば、Warn レベルには Fatal、Error および Warn レベルの各メッセージが含まれます)。

レベル	Integration Server で記録されるエントリ	例
Fatal	処理が終了してしまう障害。ユーザが対応しなければ、処理が正常に継続されません。障害が他の処理または製品に影響する可能性が非常に高くなります。	製品が設定ファイルを読み取ることができず、デフォルト設定が行われない
Error	Fatal と同じ。ただし、既存のエラー処理により、他の処理または製品には障害の影響が及びにくくなります。	入力データの間違いが原因で発生したサービスエラーにより、

レベル	Integration Server で記録されるエントリ	例
		ビジネスプロセスステップが失敗
Warn	処理の終了には至らない問題。あるいは、障害が発生しそうなことを示す不適切または異常な状態。	JMX サーバが 1 つだけ必要な場合に、複数の JMX サーバが登録されていることが検出された
Info	正常に行われたイベントの内容確認。	パッケージのロード、または接続プールの初期化
Debug	エラーを引き起こす可能性のある異常な状態または判断を記録する、コードレベルのステートメント。	オブジェクトが初期化されるはずなのに初期化されない、またはハッシュテーブルが空
Trace	通常実行時のプログラムフローと状態を記録する、コードレベルのステートメント。	メソッドの開始/終了、またはローカルオブジェクトとグローバルオブジェクトの状態
Off	製品または機能の情報をサーバログに書き込まない。	

サーバログエントリをキューに格納するかどうかの指定

デフォルトでは、Integration Server は機能によってメモリに書き込まれたログエントリをキューに格納し、バックグラウンドスレッドを使用してログエントリをサーバログに書き込みます。一度にサーバログに書き込みできる機能は 1 つだけなので、エントリをキューに格納すると、書き込みが非同期になってパフォーマンスが向上します。ただし、Integration Server が異常終了した場合は、キューに格納されているすべてのログエントリが失われます。サービスの品質を上げる場合は、エントリをキューに格納しないでください。ただし、各機能はサーバログにエントリを書き込むまで待機する必要があり、Integration Server のパフォーマンスが低下する場合があります。

`watt.server.log.queued` の値は Integration Server がサーバログエントリをキューに入れるかどうかを決定します。Integration Server でサーバログエントリをキューに入れる場合は、このパラメータを「`true`」に設定します。Integration Server でサーバログエントリをキューに入れない場合は、このパラメータを「`false`」に設定します。

注:
サーバログキューに収容可能なエントリの数を変更するには、`watt.server.serverlogQueueSize` パラメータを使用します。デフォルトのサイズは 8192 です。

サーバログのデフォルト場所の変更

ログデータを書き込む先のファイルの名前を含め、サーバログのデフォルト場所を変更できます。たとえば、Integration Server ホストマシン上の領域を節約する場合は、ログファイルの保存場所を別のマシンのディレクトリに変更できます。watt.debug.logfile サーバ設定パラメータはログファイルの場所と名前を制御します。

デフォルトで、サーバログの場所は *Integration Server_directory* \instances*instance_name*\logs\server.log です。

注:
Microservices Runtime の場合、デフォルトのサーバログの場所は *Integration Server_directory* \logs\server.log です。

サーバログの場所またはファイルを変更するには、watt.debug.logfile パラメータを変更して、サーバがサーバログ情報を書き込む先のファイルへの完全修飾パスまたは相対パスを指定します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* \instances*instance_name* に相対的なパスです。

注:
Microservices Runtime のホームディレクトリは *Integration Server_directory* です。

ディレクトリとファイル名を指定する必要があります。変更内容を有効にするには、Integration Server を再起動する必要があります。

サーバログをサイズに基づいて循環するように設定する

デフォルトでは、Integration Server は、server.log を毎日午前零時に循環します。サービスが大きなペイロードをログ記録する場合は、server.log のサイズが急激に増えます。大きな server.log ファイルは、リソースを消費するだけでなく、確認が難しくなることがあります。これを避けるために、日単位だけではなくサイズ単位で server.log ファイルを循環するように Integration Server を設定できます。

Integration Server には、サーバログのサイズ制限の指定に使用できるサーバ設定パラメータが用意されています。`watt.server.serverlogRotateSize` が有効な値に設定されている場合は、Integration Server は server.log のファイルサイズがそのサイズに到達したときまたは午前零時になったときのどちらかが最初に発生した場合に server.log を循環します。たとえば、watt.server.serverlogRotateSize が 100 KB に設定され、午前零時に server.log ファイルが 80 KB であった場合は、Integration Server は午前零時に server.log ファイルを循環します。

Integration Server が server.log ファイルを循環する場合に、Integration Server は現在のログの名前をアーカイブファイル名を使用するように変更して、新しい server.log ファイルを起動します。サイズに基づく循環である場合、アーカイブファイル名は server.log.*yyyyMMdd.n* 形式を使用します。ここで、*yyyyMMdd* はログファイルの作成日です。*n* はその日に作成されたログファイルの数です。

注:
`watt.server.serverlogRotateSize` パラメータのデフォルト値はありません。
`watt.server.serverlogRotateSize` の値が指定されない場合は、Integration Server は server.log ファイルを午前零時にのみ循環します。無効な値が指定されると、Integration Server はパラメータを -1 にリセットします。値を -1 にすると、パラメータの値を指定しない場合と同じ動作になります。

`watt.server.serverlogRotateSize` パラメータの詳細については、1095 ページの「サーバ設定パラメータ」のパラメータの説明を確認してください。

Integration Serverによって保持保持されるサーバログファイル数の制限

デフォルトで、Integration Server はアーカイブされたサーバログファイルを無期限に保持します。より冗長なログレベルを使用すると、サーバログファイルは急速に膨大または多数になることがあるため、Integration Server がファイルシステムに保持するサーバログファイルの数の制限が必要になることがあります。

Integration Server には、`watt.server.serverlogFilesToKeep` サーバ設定パラメータが用意されています。このパラメータを使用して、現在のサーバログファイルを含め、Integration Server がファイルシステムに保持するサーバログファイル数の制限を設定できます。Integration Server がサーバログファイル数の制限に達した場合、Integration Server はサーバログを循環するごとに、アーカイブ済みの最も古いログファイルデータを削除します。次に、watt.server.log.filesToKeep の値およびその結果の Integration Server の動作の例を示します。

- `watt.server.serverlogFilesToKeep` を *n* に設定した場合、Integration Server は現在のサーバログファイル (server.log) および *n-1* 個までのアーカイブ済みサーバログファイルを保持します。たとえば、watt.server.serverlogFilesToKeep を 30 に設定した場合、Integration Server は現在のサーバログファイル (server.log) および 29 個までのアーカイブ済みサーバログファイルを保持します。
- `watt.server.serverlogFilesToKeep` を 1 に設定した場合、Integration Server は現在の server.log ファイルのみ保持し、以前の server.log ファイルは保持しません。Integration Server が server.log を循環するとき、Integration Server は以前のサーバログのアーカイブファイルを作成しません。
- `watt.server.serverlogFilesToKeep` を 0 または 1 未満の値に設定した場合、Integration Server は無制限の数のサーバログファイルを保持します。

`watt.server.serverlogFilesToKeep` のデフォルト値は -1 です。これは、Integration Server が保持するサーバログファイル数に制限がないことを意味します。

`watt.server.serverlogFilesToKeep` パラメータの詳細については、1095 ページの「サーバ設定パラメータ」のパラメータの説明を確認してください。

重大な問題に関するメッセージの電子メールアドレスへの送信

デフォルトで、Integration Server はログエントリに関する通知を電子メールアドレスに一切送信しません。Integration Server は、重大な問題が発生するたびに、指定した電子メールアドレスに通知を自動的に送信するように設定できます。

重大な問題に関するメッセージを電子メールアドレスに送信するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [リソース] に移動します。
3. [**リソースの設定の編集**] をクリックします。

4. [**SMTP サーバ**] フィールドで、メッセージの送信に使用する SMTP サーバのサーバ名または IP アドレスを入力します。

5. [**ポート**] フィールドで、Integration Server が接続する、指定した SMTP サーバのポートを入力します。

6. [**転送レイヤセキュリティ**] リストから、上記で定義した SMTP サーバポートと Integration Server の通信に使用する SSL 暗号化のタイプを選択します。

選択項目	目的
[なし]	Defaultセキュアでないモードを使用して SMTP サーバと通信します。
[明示的]	Integration Server は明示的なセキュリティを使用して SMTP サーバと通信します。
[暗黙的]	Integration Server は暗黙的なセキュリティを使用して SMTP サーバと通信します。

7. [**トラストストアエイリアス**] リストから、Integration Server が Integration Server と SMTP サーバ間の信用関係の妥当性検査に使用する認証リストを含む、トラストストアのエイリアスを選択します。トラストストアエイリアスを選択しなかった場合、watt.security.trustStoreAlias プロパティに指定されているデフォルトのトラストストアエイリアスが使用されます。このプロパティの詳細については、1129 ページの「watt.security.」を参照してください。トラストストアエイリアスの詳細については、515 ページの「トラストストアエイリアスの作成」を参照してください。

8. [**内部用電子メール**] フィールドで、重大ログエントリに関するメッセージの送信先電子メールアドレスを入力します。通常は、Integration Server Administrator の電子メールアドレスを指定します。

9. [**サービス用電子メール**] フィールドで、サービスの失敗に関するメッセージの送信先電子メールアドレスを入力します。開発環境では、これらのメッセージを開発者に向けて送信します。実稼動環境では、これらのメッセージを Integration Server Administrator に向けて送信します。

10. デフォルトでは、Integration Server で使用するメッセージの文字セットは UTF-8 です。別の文字セットを使用する場合は、[**デフォルトの電子メール文字セット**] フィールドで文字セットを指定します。

11. [**変更内容の保存**] をクリックします。

    Integration Server は、指定された SMTP サーバの次のポートに接続します。

    - Integration Server が明示的な転送レイヤセキュリティを使用する、または転送レイヤセキュリティを使用しない場合はポート 25
    - Integration Server が暗黙的な転送レイヤセキュリティを使用する場合はポート 465

デフォルトは 25 です。

Integration Server はメッセージを送信するとき、デフォルトでサーバのアドレス (送信者アドレス) を Integration-Server@*localhost* に指定します。ここで *localhost* は Integration Server ホストのマシンを示します。このプロパティを変更する場合、Integration Server Administrator の [**設定**] > [拡張] ページを使用して、`watt.server.email.from` パラメータを、メッセージが使用する送信者アドレスに設定します。

注:
OAuth はリソースの電子メール通知にはサポートされていません。

ログエントリに対する追加処理の実行

ログエントリに対して追加処理を実行する場合は、イベントハンドラを作成します。たとえば、Windows イベントログなどの別のログに、サービスログエントリを送信するイベントハンドラを作成できます。詳細については、『*webMethods Integration Server Built-In Services Reference*』および『*webMethods Service Development Help*』を参照してください。

サーバログの表示

サーバログを表示するには、Integration Server Administrator で [**ログ**] > [サーバ] ページに移動します。Integration Server がデフォルトで使用するサーバログエントリの形式は次のとおりです。

time_stamptime-zone [*product_code.logging_facility.message_number.log_level*] *message_text*

ログ機能のリストを確認するには、Integration Server Administrator で [**ログ**] > [ログ設定] > [サーバロガーの詳細の表示] ページに移動します。Integration Server は JVM からタイムゾーン値を取得します。

注:
デフォルトのメッセージ形式ではログレベルが 1 文字で表示され、その値は C (Fatal)、E (Error)、W (Warn)、I (Info)、D (Debug)、T (Trace) のいずれかです。サーバログに表示されるメッセージのタイプは、ログレベル設定によって決まります。ログレベルの詳細については、237 ページの「サーバログに記録する情報の量と種類の指定」を参照してください。

次の表に、一部の製品コードを示します。

コード	意味
ISU, ISS, ISC, ISP, JBS, BAS, BAT, BAA, BAJ, BAL, BAP, BAR, BAU, BAC, BAB, BAF, BAQ	Integration Server 内部の機能/コンポーネント
ART	Adapter Runtime 機能
MNP	Mainframe パッケージ
MOD	webMethods Monitor (機能 119)

コード	意味
MON	webMethods Monitor (機能 120 Monitor DB)
SAP	SAP Adapter
TNS、TNC	Trading Networks

注:

サーバログの内容をカンマ区切り値 (csv) ファイルにエクスポートするには、 > [**CSV としてエクスポート**] をクリックします。[**表示開始行**] および [**表示するエントリ数**] フィールドの値は server_logs.csv の内容を決定します。

別のサーバログエントリ形式の使用

サーバログの形式は、Integration Server に用意されている次のような別の形式に切り替えることができます。

(logger) [*product_code.logging_facility.message_number*] *time_stamp time_zone log_level message_text*

この形式には次のような特徴があります。

- ログレベルを略さずに記述し、タイムゾーンの後に表示する。ログレベルは Fatal、Error、Warn、Info、Debug、Trace のいずれかとなります。
- *logger* フィールドには、Java ベースの log4j ログユーティリティで使用するロガーの名前を指定する。

サーバログエントリの代替形式を使用するには、watt.debug.layout サーバ設定パラメータを `new` に設定します。サーバログエントリのデフォルトの形式を使用するには、watt.debug.layout サーバ設定パラメータを `legacy` に設定します。

ログ表示の変更

Integration Server Administrator のサーバログページの表示内容を変更できます。以下の操作を実行できます。

- ログエントリで使用する日付と時刻の形式を指定する。
- ログデータをさまざまな言語で表示する。
- さまざまな表示項目を永続的に変更する。
- サーバログのさまざまな表示項目を一時的に変更する。

注:

[**ログ**] > [サーバ] ページの右上に表示される時間オプションはサーバログに適用されません。表示されるログエントリのリストを変更するには、[**表示開始行**] および [**表示するエントリ数**] を使用します。

ログエントリで使用する日付と時刻の形式の指定

Integration Server Administrator に表示されるすべてのログでは、ログエントリのデフォルト形式に *yyyy-mm-ddhh:mm:ss* を使用します。この形式は、Java クラス java.text.SimpleDateFormat がサポートする任意の形式に変更できます。

ログエントリで使用する日付と時刻の形式を指定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**ログ**] > [ログ設定] に移動します。
3. [**ロガーリスト**] で [**サーバロガー**] をクリックします。
4. [**サーバロガーの編集**] をクリックします。
5. [**形式**] ボックスで、使用する日付と時刻の形式を入力します。Java クラス java.text.SimpleDateFormat がサポートする形式であれば、いずれの形式も指定できます (yyyy-MM-dd HH:mm:ss z など)。
6. [**変更内容の保存**] をクリックします。

注:
`watt.server.dateStampFmt` サーバ設定パラメータを使用して、ログエントリの日付と時刻の形式を変更することもできます。このパラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

ログデータのさまざまな言語での表示

ここでは、ファイルに保存されたログデータのみに適用される内容について説明します。

ログデータを英語以外の言語で表示する場合は、テキストエディタまたはコマンドシェルを設定する必要があります。Integration Server によるファイルへの書き込みは、Unicode UTF-8 エンコーディングで行われます。これらのファイルには、Byte Order Mark (BOM) の Unicode 文字 U+FEFF が含まれていません。米国英語以外で書き込まれたログエントリなど、非 ASCII データがファイルに含まれている場合は、ログエントリを表示できるように、テキストエディタまたはコマンドシェルで使用する文字エンコーディングを設定する必要があります。

UNIX システムでは、ロケール設定 (LC_ALL) を適切な UTF-8 エンコードのロケールに変更して文字エンコーディングを設定します。たとえば、Solaris システムのテキストエディタまたはコマンドシェルで日本語の文字を表示するには、ロケール設定を ja_JP.UTF-8 に変更します。

Windows システムでは、ファイルに BOM 文字が含まれていないため、メモ帳などのテキストエディタで UTF-8 エンコーディングが正しく検出されない場合があります。ファイルを表示するには、エンコーディングを手動で設定する必要があります。コマンドシェルでログを表示するには、コマンド `chcp 65001` を使用します。

注:
文字エンコーディングを変更すると監査ログにも影響があります (『*webMethods Audit Logging Guide*』を参照)。

すべてのログの表示を永続的に変更

デフォルトでは、Integration Server Administrator のすべてのログに表示されるログエントリの数は 35、リフレッシュ間隔は 90 秒です。これらのデフォルト値は変更が可能です。

ログの表示を変更するときは、以下の点に留意してください。

- 表示されるエントリの数を増やしすぎると、システムのパフォーマンスが低下する可能性がある。
- リフレッシュ間隔を減らしすぎると、システムのパフォーマンスが低下する可能性があります。
- 表示されるエントリ数を、`watt.server.log.alertMaxEntries` 設定パラメータで設定したしきい値よりも大きい値にすると、Integration Server Administrator により警告が表示されます。
- ログ表示を変更すると監査ログに影響があります (『*webMethods Audit Logging Guide*』を参照)。

次の表では、ログ表示を制御するサーバ設定パラメータを示します。

変更対象	使用するサーバ設定パラメータ
[**表示するエントリ数**] フィールドのデフォルト	`watt.server.log.maxEntries` [**表示するエントリ数**] に設定された値が、`watt.server.log.alertMaxEntries` に指定された値よりも大きい場合、ログエントリがリフレッシュされるたびに、Integration Server Administrator によりメッセージが表示され、要求されたエントリ数がしきい値を超えており、さらに多くのエントリを表示しようとすると、Integration Server のパフォーマンスに影響が及ぶ可能性があると警告されます。メッセージでは、要求されたエントリ数の表示を確認するように指示されます。
ログエントリのリフレッシュ間隔	`watt.server.log.refreshInterval`
ログエントリの日付と時刻の形式	`watt.server.dateStampFmt`

サーバログ表示の一時的な変更

サーバログの表示を一時的に変更するには、ログ表示ページの上部にある [**ログの表示制御**] 領域で [**リフレッシュ**] をクリックします。この変更は、設定を再度変更するか、または Integration Server をシャットダウンするかのいずれかが行われるまで継続します。

セッションのログレベルとサーバログの場所の変更

特定のオプションを指定してコマンドラインから Integration Server を起動すると、特定の Integration Server セッションのログレベルプロパティ設定、サーバログの場所またはその両方を変更できます。

セッションのログレベルとサーバログの場所を変更するには

1. コマンドラインに次のコマンドを入力して、サーバのホームディレクトリに切り替えます。

   `cd` *Integration Server_directory* `\profiles\IS_`*instance_name*

   注:
   Microservices Runtime のホームディレクトリは *Integration Server_directory* です。

2. 次のコマンドを入力して Integration Server を起動します。

   Windows の場合: `bin\startup.bat [-debug` *level*`] [-log {`*filename* `| none | both}]`

   UNIX の場合: `bin/startup.sh [-debug` *level*`] [-log {`*filename* `| none | both}]`

   これらのコマンドのオプションを次に示します。

オプション	説明
level	現行セッションの [**ログ**] > [ログ設定] > [サーバロガー] > [サーバロガーの詳細の表示] ページに表示されるすべての製品および機能で記録される情報の量。指定可能な値については、238 ページの「ログレベル」を参照してください。 現行セッション中のみ、[**ログ**] > [ログ設定] > [サーバロガー] > [サーバロガーの詳細の表示] ページおよび watt.debug.level プロパティで指定された値がこのオプションによって変更されます。
filename	現行セッションのサーバログを書き込むファイルの完全修飾パスまたは相対パス。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* \instances*instance_name* に相対的なパスです。 **注:** Microservices Runtime のホームディレクトリは *Integration Server_directory* です。

オプション	説明
	filename にはディレクトリとファイル名を指定する必要があります。 現行セッション中のみ、watt.debug.logfile プロパティで指定された値が *filename* オプションによって変更されます。
none	サーバログをコンソール (STDOUT) に表示します。 現行セッション中のみ、watt.debug.logfile プロパティで指定された値が none オプションによって変更されます。 **注:** STDOUT に書き込まれているサーバログメッセージは “ISSERVER” 識別子を含み、サーバログメッセージをコンソールに書き込まれた他のメッセージから区別するのに役立ちます。
both	サーバログ情報をコンピュータ画面 (STDOUT) および watt.debug.logfile パラメータで指定された先に書き込みます。

注:
-log のスイッチ値である none または both は、Microservices Runtime または Microservices Runtime ライセンスを備えている Integration Server が設定変数ログを書き込む場所も指定します。Microservices Runtime はファイル名の値を無視します。ファイル名を指定した場合、Microservices Runtime は設定変数ログファイルを以下の場所だけに書き込みます。 *Integration Server_directory* /instances/*instanceName*/logs/configurationvariables.log

グローバリゼーションとサーバログ

ある webMethods 製品に webMethods Language Pack が用意されており、その製品のオペレーティング環境で使用する言語に Language Pack が対応している場合は、オペレーティングシステムで使用する言語でログエントリが書き込まれます。製品に Language Pack が用意されていないか、またはオペレーティングシステムで使用する言語に対応していない場合は、米国英語でログエントリが書き込まれます。

オペレーティング環境の言語に日本語を使用しているとします。Integration Server には日本語 Language Pack を含む Language Pack がインストールされているため、Integration Server のログエントリは日本語で保存されます。Trading Networks には日本語 Language Pack がインストールされていないため、Integration Server は Trading Networks のログエントリを米国英語で保存します。

注:
Language Pack がインストールされていない webMethods 製品で米国英語が使用されている場合でも、Integration Server は、データベースドライバまたはアダプタリソースなどの外部ソースからのログエントリを、その製品のオペレーティング環境で使用する言語で保存できます。

12 統計ログ

- 統計ログについて 250
- 統計ログの表示 252
- 統計ログに影響するサーバ設定パラメータ 253

統計ログについて

統計ログには、Integration Server のアクティビティの監視に使用できる情報が含まれます。ログには、サービス要求、セッションの使用状況、サーバリソース、および JVM スレッドの使用状況に関するデータが含まれます。統計ポーリング間隔が経過するごとに、Integration Server は統計情報を取得し書き込みます。New Requests、Completed Requests などの統計情報はポーリング間隔ごとであり、Max Running Service Threads などの他の情報はピークデータを表します。統計ログに書き込まれるほとんどのデータは [**サーバ**] > [統計情報] ページおよびダッシュボードにも表示されます。

Integration Server が統計ログのデータを収集する頻度は、watt.server.stats.pollTime サーバ設定パラメータの値によって決定されます。デフォルトで、このパラメータは 60 秒後ごとに統計情報をポーリングするよう設定されています。

統計ログのサービス要求情報にはトップレベルサービス要求のみ含まれます。トップレベルサービス要求はクライアント要求、トリガー、またはスケジュールされているタスクによって呼び出されるサービスです。ただし、Integration Server は、トップレベルサービスとして呼び出される内部サービスに関する統計情報を統計ログに記録しません。内部サービスは、名前が Wm で始まるパッケージに属するサービスです。watt.server.stats.packages.exclude サーバ設定パラメータを使用して、統計情報から除外されるサービスを変更できます。

Integration Server 10.7 以降、統計ログのデフォルトの名前と形式は stats.csv という名前の csv (カンマ区切り) ファイル です。csv 形式によって、Integration Server の統計情報はスプレッドシートプログラムに簡単にインポートできます。10.7 より前の Integration Server では、統計ログに 16 進値で、空白を十分に使用したわかりやすい書式設定でデータが書き込まれました。watt.server.stats.logfile.csv パラメータを使用して、統計ログの形式を変更できます。

以下に、Microsoft Excel に表示される csv 形式の統計ログの例を示します。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Time	Total Memory	Free Memory	Current JVM Threads	Max JVM Thread	Total Sessions	Max Sessions	Session Average T	Running Service Threads	Max Running Service Threads	Request Average Time	New Requests	Completed Requests
2	2021-01-15 22:23:31 EST	530579456	223209544	167	167	0	1	0	0	1	0	0	0
3	2021-01-15 22:24:31 EST	530579456	193262192	167	167	0	1	0	0	1	0	0	0
4	2021-01-15 22:25:31 EST	530579456	166705536	167	167	0	1	0	0	1	0	0	0
5	2021-01-15 22:26:31 EST	530579456	139265192	167	167	0	1	0	0	1	0	0	0
6	2021-01-15 22:27:31 EST	530579456	112743736	167	167	0	1	0	0	1	0	0	0
7	2021-01-15 22:28:31 EST	532676608	280688480	167	167	0	1	0	0	1	0	0	0
8	2021-01-15 22:29:31 EST	532676608	253459344	167	167	0	1	0	0	1	0	0	0
9	2021-01-15 22:30:31 EST	532676608	226590472	167	167	0	1	0	0	1	0	0	0
10	2021-01-15 22:31:31 EST	532676608	199973184	167	167	0	1	0	0	1	0	0	0
11	2021-01-15 22:32:31 EST	532676608	173048928	167	167	0	1	0	0	1	0	0	0
12	2021-01-15 22:33:31 EST	513277952	327197432	167	167	0	1	0	0	1	0	0	0
13	2021-01-15 22:34:31 EST	513277952	297669216	167	167	0	1	0	0	1	0	0	0
14	2021-01-15 22:35:31 EST	513277952	271806752	167	167	0	1	0	0	1	0	0	0
15	2021-01-15 22:36:31 EST	513277952	243912928	167	167	0	1	0	0	1	0	0	0
16	2021-01-15 22:37:31 EST	513277952	218095328	167	167	0	1	0	0	1	0	0	0
17	2021-01-15 22:38:31 EST	513277952	123681784	167	167	0	1	0	0	1	0	0	0
18	2021-01-15 22:39:31 EST	517472256	337498128	167	167	0	1	0	0	1	0	0	0
19	2021-01-15 22:40:31 EST	517472256	310966632	167	167	0	1	0	0	1	0	0	0
20	2021-01-15 22:41:31 EST	517472256	285168840	167	167	0	1	0	0	1	0	0	0
21	2021-01-15 22:42:31 EST	517472256	257586960	167	167	0	1	0	0	1	0	0	0

以下に、16 進形式の統計ログの例を示します。

```
2021-01-15 21:11:53 EST ---( Start )---
2021-01-15 21:11:53 EST TotalMem         FreeMem          CurT MaxT SSNs SSNx SSNAvg   REQs REQx REQAvg   StartReq EndReq
2021-01-15 21:12:53 EST 0000000029180000 000000001762baf0 0093 0093 0000 0001 0000003f 0000 0001 0000003f 0000000f 0000000f

2021-01-15 21:13:53 EST 0000000029180000 0000000014c499d0 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:14:53 EST 0000000029180000 00000000125d1c48 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:15:53 EST 0000000028300000 000000001cceec08 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:16:53 EST 0000000028300000 000000001a1ea918 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:17:53 EST 0000000028300000 0000000017c21070 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:18:53 EST 0000000028300000 0000000015670288 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:19:53 EST 0000000028300000 000000001311e248 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:20:53 EST 0000000028d00000 000000001f79a110 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:21:53 EST 0000000028d00000 000000001caa1cd0 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:22:53 EST 0000000028d00000 000000001a467ac0 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000

2021-01-15 21:23:53 EST 0000000028d00000 0000000017fbaad0 0093 0093 0000 0001 00000000 0000 0001 00000000 00000000 00000000
```

Integration Server は統計ログを *Integration Server_directory* \instances*instanceName*\logs\stats.csv に保存します。watt.server.stats.logfile サーバ設定パラメータを使用して、統計ログの場所と名前を変更できます。

Integration Server の起動後、最初のポーリング間隔後に統計ログが作成されます。その後、Integration Server は watt.server.statsLogRotateInterval サーバ設定パラメータ (分単位) によって指定された間隔で統計ログを循環します。たとえば、watt.server.statsLogRotateInterval を 60 分に設定し、Integration Server が午後 1:30 に起動された場合、Integration Server は午後 2:30、午後 3:30、午後 4:30 などに stats.log を循環します。この動作の例外として、ログ再利用時間が Integration Server の起動時刻から午前零時を超える場合、Integration Server は最初に午前零時に統計ログを循環します。その後、Integration Server は指定間隔でログを循環します。たとえば、watt.server.statsLogRotateInterval が 300 分 (5 時間) に設定され、Integration Server が午後 9:30 に起動された場合、Integration Server は午前零時に統計ログファイルを循環し、その後は 5 時間ごとに stats.log を循環します。Integration Server は常に午前零時に統計ログファイルを循環します。

また、サイズに基づいて統計ログを循環するよう設定することもできます。特定のサイズに達したら統計ログを循環すると、ログファイルの使いにくさを避けることができます。Integration Server が統計ログをロールオーバーするファイルサイズを指定するには、watt.server.stats.logRotateSize サーバ設定パラメータを使用します。デフォルトでは、Integration Server はサイズに基づいて統計ログを循環しません。

サイズに基づいて統計ログを循環するよう設定した場合、Integration Server はサイズと時間 (いずれか早い方) に基づいて統計ログをロールオーバーします。

統計ログを循環する場合、Integration Server は以前の統計ログの名前を stats*yyyymmdd_hhmmss*.csv (または、統計ログが 16 進形式である場合、stats*yyyymmdd_hhmmss*.log) に変更します。デフォルトで、Integration Server は無制限の数のアーカイブされた統計ログを保持します。watt.server.stats.logFilesToKeep サーバ設定パラメータを使用して、Integration Server で保持されるアーカイブされたファイルの数を制限できます。

統計ログの表示

統計ログには、システムリソースおよび Integration Server リソースの消費量に関する統計情報が含まれます。Integration Server は統計ログを *Integration Server_directory* \instances*instanceName*\logs ディレクトリに保存します。

他のログと異なり、Integration Server Administrator の [**ログ**] メニューの下に統計ログを表示することはできません。ただし、多くの統計ログは [**サーバ**] > [統計情報] ページまたはダッシュボードにも表示されます。

watt.server.stats.logfile.csv パラメータの値に従って、統計ログは csv (カンマ区切り) ファイルまたは 16 進です。統計ログが csv ファイルである場合、フィールド値は 16 進数で表示されます。統計ログには、形式に関係なく同じデータが含まれます。

統計ログを表示する場合は、以下の情報に留意してください。

- 要求数 (Request Average Time、New Requests、および Completed Requests) には、トップレベルの要求のみ含まれます。トップレベルサービス要求はクライアント要求、トリガー、またはスケジュールされているタスクによって呼び出されるサービスです。トップレベルサービスを呼び出すと他のサービスが呼び出される場合、そのサービス呼び出しチェーン全体が 1 つの要求と見なされます。
  - サーキットブレーカーのサーキットのオープンサービスとして呼び出されるサービスは、トップレベルサービス要求としては扱われません。
  - トリガーのリソース監視サービスとして、またはトリガーによる重複抑制処理のドキュメントリゾルバサービスとして呼び出されるサービスは、トップレベルサービスと見なされます。
- デフォルトでは、Integration Server は要求数に内部サービスに関するデータを取得しません。これは、内部サービスがトップレベルサービスとして呼び出される場合も当てはまります。内部サービスは名前が Wm で始まるパッケージのサービスです。watt.server.stats.packages.exclude サーバ設定パラメータを使用して、統計情報から除外されるサービスを変更できます。
- Integration Server による JVM スレッドの使用数を計算する場合、Integration Server で使用されたシステムスレッドのみ含まれます。この数には JVM で開始された他のスレッドまたは JVM で開始されたプロセスは含まれません。

次の表では、csv 形式のフィールド、16 進形式で使用される列名など、統計ログに書き込まれる情報を示して説明しています。

CSV フィールド名	16 進フィールド名	説明
Time	Time	Integration Server がログエントリを統計ログに書き込んだタイムスタンプ。
Total Memory	TotalMem	Integration Server を実行する JVM で使用できるメモリの総容量。

CSV フィールド名	16 進フィールド名	説明
Free Memory	FreeMem	Integration Server を実行する JVM で使用できるメモリの空き容量。
Current JVM Threads	CurT	このポーリング間隔で Integration Server によって開始された JVM スレッドの数。
Max JVM Threads	MaxT	Integration Server の起動時から Integration Server によって使用された JVM スレッドの最大数。
Total Sessions	SSNs	ポーリング間隔の最後に Integration Server で並行して実行されているアクティブなセッションの合計数。
Max Sessions	SSNx	Integration Server の起動以降、ポーリング間隔の最後に Integration Server で並行して実行されているセッションの最大数。
Session Average Time	SSNAvg	ポーリング間隔中に完了したセッションの平均時間 (ミリ秒単位)。セッションの平均時間の計算では、ポーリング間隔中に完了したすべてのセッションの長さの合計をポーリング間隔中に完了したセッションの数で割ります。
Running Service Threads	REQs	ポーリング間隔でトップレベルサービス要求のためにアクティブに実行されているサービススレッド数。トップレベルサービスからの要求により生成されたサービススレッドはこの数に含まれません。
Max Running Service Threads	REQx	Integration Server の起動以降、アクティブに実行されているサービススレッドの最大数。トップレベルサービスからの要求により生成されたサービススレッドはこの数に含まれません。
Request Average Time	REQAvg	ポーリング間隔中に完了した要求の平均時間 (ミリ秒単位)。要求の平均時間の計算では、ポーリング間隔中に完了したすべての要求の合計時間をポーリング間隔中に完了した要求の数で割ります。
New Requests	StartReq	ポーリング間隔中に開始された要求の数。
Completed Requests	EndReq	このポーリング間隔中に完了された要求の数。

統計ログに影響するサーバ設定パラメータ

Integration Server には、統計ログの出力またはログの循環頻度の調整に使用できるサーバ設定パラメータが含まれます。これらのパラメータは、[**サーバ**] > [統計情報] ページに表示される収集情報にも影響を与えることができます。

次の表では、統計ログで使用されるサーバ設定パラメータを示します。

サーバ設定パラメータ	説明
watt.server.stats.logfile	統計ログの名前と場所。
watt.server.stats.logfile.csv	統計ログの形式。「true」を設定した場合、Integration Server はカンマ区切りファイル stats.csv に統計ログを書き込みます。「false」を設定した場合、Integration Server は 16 進形式ファイル statistics.log に統計ログを書き込みます。
watt.server.stats.logFilesToKeep	現在のログファイルを含め、Integration Server がファイルシステムに保持する統計ログファイルの数。
watt.server.statsLogRotateInterval	Integration Server が統計ログを循環するまでに経過する時間 (分)。
watt.server.stats.logRotateSize	Integration Server が統計ログをロールオーバーするファイルサイズ。
watt.server.stats.packages.exclude	統計ログ、[**サーバ**] > [統計情報] ページ、[**サーバ**] > [サービスの使用状況] ページ、およびダッシュボードの [使用状況] タブにサービスが含めないパッケージ。
watt.server.stats.pollTime	統計情報のログの更新間隔の秒数。

上記のサーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

13 webMethods メッセージングのための Integration Server の設定

- webMethods メッセージングの概要 256
- メッセージング接続エイリアスの使用 256
- Universal Messaging Server への接続の認証 281
- Broker 接続へのキープアライブモードの指定 283
- Integration Server のプライマリポートが変更された場合の Broker クライアントの同期 . 285
- ドキュメントストアの設定 286
- Integration Server の非クラスタグループとの負荷分散 294

webMethods メッセージングの概要

webMethods メッセージングとは、ドキュメント (メッセージとも呼ばれる) を webMethods プラットフォームで送受信することを意味する包括的な用語です。Integration Server のコンテキストで、webMethods メッセージングには、ネイティブ IData 形式のドキュメントを一方の Integration Server から、または Integration Server で稼働するアプリケーションから、webMethods メッセージングプロバイダへパブリッシュすることが含まれます。webMethods メッセージングプロバイダには Software AG Universal Messaging または webMethods Broker があり、ドキュメントを抽出および処理する他方の Integration Server のサブスクライバーにそのドキュメントをルーティングします。webMethods メッセージはネイティブメッセージングとしても参照されます。

Integration Server を使用して webMethods メッセージングのパブリッシュおよびサブスクライブソリューションを作成するには、以下のコンポーネントが必要です。

- パブリッシュ可能なドキュメントタイプ。パブリッシュ可能なドキュメントの内容、構造、およびプロパティを定義します。
- パブリッシュ可能なドキュメントタイプのインスタンスをパブリッシュするサービス。
- パブリッシュ可能なドキュメントタイプにサブスクライブし、かつパブリッシュ可能なドキュメントタイプのインスタンスを処理するサービスを指定する webMethods messaging trigger。
- サブスクライバーへのドキュメントの受信およびルーティングを処理する webMethods メッセージングプロバイダ。

Integration Server を webMethods メッセージング用に設定するには、次の作業を実行する必要があります。

- webMethods メッセージングプロバイダへの接続を設定する 1 つ以上のエイリアスを作成します。詳細については、256 ページの「メッセージング接続エイリアスの使用」を参照してください。
- パブリッシュの直前または抽出の直後にドキュメントを保持するドキュメントストアを設定します。詳細については、286 ページの「ドキュメントストアの設定」を参照してください。
- Software AG Designer の Service Development パースペクティブを使用して、パブリッシュ可能なドキュメントタイプ、webMethods messaging trigger、およびサービスを作成します。詳細については、*webMethods Service Development Help*を参照してください。

注:
webMethods Broker は廃止されました。

メッセージング接続エイリアスの使用

メッセージング接続エイリアスは、Integration Server と webMethods メッセージングプロバイダの間で接続を確立するために必要な設定を定義します。Universal Messaging や Broker を webMethods メッセージングプロバイダとして使用することができます。使用するメッセージングプロバイダごとに、メッセージング接続エイリアスを作成する必要があります。

Integration Server に Broker へ接続するメッセージング接続エイリアスを 1 つのみ設定できる場合は、Universal Messaging サーバに接続するメッセージング接続エイリアスを複数使用することができます。

メッセージング接続エイリアスはパブリッシュ可能なドキュメントタイプに割り当てます。パブリッシュサービスは、割り当てられたメッセージング接続エイリアスを使用して、そのドキュメントタイプのインスタンスをメッセージングプロバイダにパブリッシュします。webMethods messaging trigger はメッセージング接続エイリアスを使用して、メッセージングプロバイダからパブリッシュされたドキュメントを抽出します。

事前定義済みのメッセージング接続エイリアス

Integration Server には、Integration Server が初回起動時に作成する事前定義済みのメッセージング接続エイリアスが含まれます。

次の表に、Integration Server 用に使用可能な事前定義済みメッセージング接続エイリアスを示します。

メッセージング接続エイリアス	説明
IS_BROKER_CONNECTION	Broker への接続を確立するために必要な設定情報が含まれる、メッセージング接続エイリアス。 **注:** このエイリアスを使用したか Broker への接続を設定していた Integration Server の以前のバージョンから Integration Server 10.2 以降に移行した場合にのみ、このメッセージング接続エイリアスは表示されます。
IS_DES_CONNECTION	Digital Event Servicesとのイベントの送受信のための Universal Messaging サーバとの接続の確立に使用されるメッセージング接続エイリアス。Digital Event Servicesにイベントとしてパブリッシュされるパブリッシュ可能なドキュメントタイプにこのメッセージング接続エイリアスを割り当てます。 IS_DES_CONNECTION エイリアスは削除できません。
IS_LOCAL_CONNECTION	ローカルにのみパブリッシュされるドキュメントタイプで使用されるメッセージ接続エイリアス。ローカルパブリッシュでは、ドキュメントは Integration Server 内でパブリッシュおよび受信されます。ローカルパブリッシュの場合、ドキュメントは Integration Server 内に残ります。webMethods メッセージングプロバイダからの関連はありません。 **注:** Integration Server バージョン 9.9 以降に移行して、デフォルトの接続エイリアスを指定していない Integration Server を移行している場合、Integration Server は IS_LOCAL_CONNECTION エイリアスを DEFAULT を指定

メッセージング接続エイリアス	説明
	していないパブリッシュ可能なドキュメントタイプに割り当てるか、メッセージング接続エイリアスを指定しません。
IS_UM_CONNECTION	Universal Messaging サーバへの接続を確立するために必要な設定情報が含まれるメッセージング接続エイリアス。

Broker 接続エイリアスの作成

Broker 接続エイリアスは、Broker への接続を確立するための設定情報が含まれる webMethods メッセージング接続エイリアスです。各 Integration Server にはそれぞれ 1 つの Broker 接続エイリアスしか設定できません。

注:
Integration Server バージョン 10.11 は、Broker 9.6 または Broker 10.5 に接続されているときと同じように動作します。

Integration Server の以前のバージョンから Integration Server 10.2 以降に移行した場合、Broker 接続エイリアス IS_BROKER_CONNECTION が既に存在する可能性があります。または、Integration Server の以前のバージョンから Integration Server 9.5 SP1 以降に移行していて、以前の Integration Server で Broker への接続を設定していた場合、IS_BROKER_CONNECTION エイリアスには既存の Broker 設定情報が使用されます。このエイリアスが存在する場合は、別の Broker 接続エイリアスを作成する前に削除する必要があります。または、必要な設定情報を含むように IS_BROKER_CONNECTION を編集することができます。

注:
webMethods Broker は廃止されました。

Broker 接続エイリアスを作成するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods の設定] に移動します。
3. [**Broker 接続エイリアスの作成**] をクリックします。

   Broker 接続エイリアスが既に存在する場合、Integration Server Administrator は存在できる Broker 接続エイリアスは一度に 1 つのみであるというメッセージを表示します。
4. [**全般設定**] で、以下を指定します。

フィールド	説明
[接続エイリアス名]	メッセージング接続エイリアスの一意の名前。

フィールド	説明
[説明]	メッセージング接続エイリアスの説明。
[クライアントプリフィックス]	Integration Server が Broker を識別するためのストリング。 Broker Manager では、サーバに対して作成されるクライアントごとにこのプリフィックスが表示されます(Broker は、接続するサーバごとに複数のクライアントを作成します)。
[クライアントプリフィックスの共有(共有されたメッセージングプロバイダオブジェクトが消去されるのを防止)]	Integration Server が複数の Integration Server とクライアントプリフィックスを共有し、Broker のこのエイリアスと関連付けられた共有オブジェクトに対する削除を含む自動更新を行わないかどうか。Broker の共有オブジェクトに対する自動更新を行わない場合、[**クライアントプリフィックスの共有**] チェックボックスをオフのままにします。 [**クライアントプリフィックスの共有**] チェックボックスをオンにした場合、Broker のオブジェクトを手動で更新する必要があります。たとえば、Integration Server のトリガーを削除した場合、Broker の関連するクライアントキューを手動で削除する必要があります。 Integration Server が次の場合、[**クライアントプリフィックスの共有**] をオンにする必要があります。 ■ クラスタに属し、Broker のこのエイリアスと関連付けられた共有オブジェクトに対する自動更新を行わない。 ■ 負荷分散された方法で動作する Integration Server の非クラスタグループに属し、Broker のこのエイリアスと関連付けられた共有オブジェクトに対する自動更新を行わない。 負荷分散される方法で Broker からメッセージを受信するように Integration Server を設定する詳細については、294 ページの「Integration Server の非クラスタグループとの負荷分散」を参照してください。

5. [**接続設定**] で、以下を指定します。

フィールド	説明
[Broker ホスト]	Broker Server がインストールされているマシンの名前 (*DNSname*:*port* または *ipaddress*:*port*)。
[Broker 名]	Broker Server に定義されている Broker の名前。デフォルトの名前は「**Broker #1**」です。
[クライアントグループ]	この Broker の接続先となる Integration Server クライアントグループ。単一の Broker 上で 1 つまたは複数のクライアント (この場合は Integration Server) に割り当てられた、プロパティとアクセス権限のセットを定義します。指定されたクライアントグループが存在しない

フィールド	説明
	場合、Integration Server は、初回の接続確立時に、指定された Broker 上でクライアントグループを作成します。 **重要:** Broker では、パブリッシュ可能およびサブスクライブ可能の許可はクライアントグループ間で共有されません。1 つのクライアントグループから別のクライアントグループに Integration Server を切り替えたときには、Integration Server を再起動し、パブリッシュ済みのすべてのドキュメントタイプを Broker と同期させる必要があります。次に、サーバをシャットダウンし、My webMethods を使用して、クライアントグループを変更したサーバ用に作成されたすべての Broker クライアントを削除する必要があります。その後、変更したクライアントグループのサーバを再起動します。

6. [**クライアント認証の設定**] で、以下を指定します。

フィールド	説明
[クライアント認証]	Integration Server クライアントが Broker に接続するために使用する認証のタイプ。以下のいずれかのオプションを選択します。 ■ [**なし**]匿名接続を許可するように Broker を設定する場合は、このオプションを選択します。 ■ [**ユーザ名/パスワード**]Broker が基本的なクライアント認証を使用する場合は、このオプションを選択します。このオプションを選択した場合は、[**ユーザ名**] および [**パスワード**] フィールドでクライアントが使用するユーザ名およびパスワードを指定します。 ■ [**SSL**]SSL ポートを使用して Integration Server が Broker に接続する場合は、このオプションを選択します。このオプションを選択した場合は、以下の情報を指定して SSL パラメータを設定します。

フィールド	説明
[キーストア]	この Integration Server のキーストアファイルの完全パス。キーストアファイルは、SSL 認証に必要となるクレデンシャル (秘密鍵/署名付き認証) を含みます。Broker Server が SSL 接続を必要とする場合は、Integration Server クライアントを Broker Server に対して認証するために、このファイル内の情報が使用されます。 Integration Server のキーストアファイルは、Integration Server が常駐するマシンに保存されます。

フィールド	説明
[キーストアタイプ]	Integration Server のキーストアファイルのファイルタイプ。ファイルタイプは、「**PKCS12**」または「**JKS**」のいずれかになります。
[キーストアのパスワード]	Integration Server のキーストアファイル内の SSL 認証にアクセスするために必要なパスワード。

7. [**暗号化設定**] で、以下を指定します。

フィールド	説明
[暗号化]	Integration Server と Broker の間の接続を暗号化するかどうかの指定。 **注:** [**クライアント認証**] を [**SSL**] に設定した場合、[**暗号化**] を [**はい**] に設定する必要があります。 [**暗号化**] パラメータに [**はい**] を選択した場合、以下のトラストストアパラメータを設定します。

フィールド	説明
[トラストストア]	Integration Server クライアントのトラストストアファイルの完全パス。トラストストアファイルは、SSL 認証の署名責任を負う認証局の信用のあるルート認証を含みます。SSL 接続を確立するには、トラストストアファイルから、キーストアに保存された SSL 認証の有効で信用のあるルートにアクセスできる必要があります。 Integration Server のトラストストアファイルは、Integration Server が常駐するマシンに保存されます。
[トラストストアタイプ]	Integration Server のトラストストアファイルのファイルタイプ。ファイルタイプは「**JKS**」です。

8. [**変更内容の保存**] をクリックします。

   Integration Server は、Broker 接続エイリアスを作成します。

9. [**webMethods メッセージングの設定に戻る**] をクリックします。

10. Broker 接続エイリアスを有効にします。

11. Integration Server を再起動します。

Broker および Broker に対して SSL を設定する方法の詳細については、『*Administering webMethods Broker*』を参照してください。

注:
この接続エイリアスによって使用される Broker が利用できないときにパブリッシュされるドキュメントを含めるように送信ドキュメントストア (クライアントサイドキューとも呼ばれる) を設定することができます。詳細については、以下を参照してください。291 ページの「送信ドキュメントストアについて」

Universal Messaging 接続エイリアスの作成

Universal Messaging 接続エイリアスは、Universal Messaging サーバへの接続を確立するための設定情報が含まれる webMethods メッセージング接続エイリアスです。各 Integration Server にはそれぞれ複数の Universal Messaging 接続エイリアスを設定できます。

Integration Server が Universal Messaging 接続エイリアスを使用して Universal Messaging サーバのセキュアソケットプロトコルポートに接続する場合は、接続エイリアスを SSL を使用するように設定する必要があります。エイリアスでトラストストアと場合によってはキーストア情報を指定します。これにより、Integration Server と Universal Messaging との間の通信は SSL (Secure Socket Layer) を介してセキュリティ保護されます。Universal Messaging ポートのインタフェースプロトコルとして NSPS または NHPS を指定している場合は、ポートで SSL が使用されます。Universal Messaging ポートの設定の仕方によっては、クライアント認証の指定が必要になることがあります。

- Universal Messaging ポートで NSPS または NHPS を指定しており、ポートに対して [**Enable Client Cert Validation**] オプションがオンになっていない場合、Universal Messaging 接続エイリアスの設定中にトラストストアエイリアスを指定する必要があります。
- Universal Messaging ポートで NSPS または NHPS を指定しており、ポートに対して [**Enable Client Cert Validation**] オプションがオンになっている場合、Universal Messaging 接続エイリアスの設定中にトラストストアエイリアス、キーストアエイリアス、およびキーエイリアスを指定する必要があります。

注:
ポート用に選択されたプロトコルおよび [**Enable Client Cert Validation**] オプションを表示するには、Universal Messaging Enterprise Manager を使用します。

Universal Messaging 接続エイリアスが Universal Messaging のセキュアポートへの接続を確立する場合、エイリアスの設定を完了する前に以下の手順に従う必要があります。

- Universal Messaging サーバの認証用の認証局 (CA) を含むトラストストアを作成します。トラストストア用のトラストストアエイリアスを作成します。トラストストアエイリアスの作成の詳細については、515 ページの「トラストストアエイリアスの作成」を参照してください。
- Universal Messaging ポートがクライアント認証の妥当性検査を行う場合、以下の手順に従います。

- Integration Server が Universal Messaging との接続に使用するクライアント認証を含むキーストアを作成します。キーストアのキーストアエイリアスを作成し、Universal Messaging ポートを安全に接続するためのプライベートキーを含むキーのキーエイリアスを指定します。キーストアエイリアスの作成の詳細については、513 ページの「キーストアエイリアスの作成」を参照してください。
- Universal Messaging で使用されるトラストストアに Integration Server で使用される証明書の CA が含まれていることを検証します。

注:
Universal Messaging クライアントライブラリを最新の状態にしておきます。クライアントライブラリを更新する Universal Messaging の修正をインストールする場合は、更新された Universal Messaging クライアントライブラリファイルを、Integration Server が使用する *Software AG_directory* /common/lib ディレクトリにコピーして下さい。

Universal Messaging 接続エイリアスを作成するには、以下の手順に従います。

Universal Messaging 接続エイリアスを作成するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods の設定] に移動します。
3. [**Universal Messaging 接続エイリアスの作成**] をクリックします。
4. [**全般設定**] で、以下を指定します。

フィールド	説明
[接続エイリアス名]	メッセージング接続エイリアスの一意の名前。
[説明]	メッセージング接続エイリアスの説明。
[クライアントプリフィックス]	Integration Server が Universal Messaging を識別するためのストリング。 使用する Integration Server がクラスタの一部となっている場合は、必ずクラスタ内のすべてのサーバで同じクライアントプリフィックスを使用してください。
[クライアントプリフィックスの共有 (共有メッセージングプロバイダオブジェクトを削除しない)]	Integration Server が複数の Integration Server とクライアントプリフィックスを共有し、Universal Messaging サーバのこのエイリアスと関連付けられた共有オブジェクトに対する削除を含む自動更新を行わないかどうか。Universal Messaging の共有オブジェクトに対する自動更新を行わない場合、チェックボックスをオフのままにします。 [**クライアントプリフィックスの共有**] チェックボックスをオンにした場合、Universal Messaging のオブジェクトを手動で更新する必要があります。たとえば、Integration Server のトリガーを削除した場合、

フィールド	説明
	Universal Messaging の関連するチャネルを手動で削除する必要があります。 Integration Server が次の場合、[**クライアントプリフィックスの共有**] をオンにする必要があります。 ■ クラスタに属し、Universal Messaging サーバのこのエイリアスと関連付けられた共有オブジェクトに対する自動更新を行わない。 ■ 負荷分散された方法で動作する Integration Server の非クラスタグループに属し、Universal Messaging サーバのこのエイリアスと関連付けられた共有オブジェクトに対する自動更新を行わない。 負荷分散される方法で Universal Messaging からメッセージを受信するように Integration Server を設定する詳細については、294 ページの「Integration Server の非クラスタグループとの負荷分散」を参照してください。 ■ Universal Messaging 接続エイリアスは、クラスタグループまたは非クラスタグループの複数の Integration Server のロガーによって、Universal Messaging キューのログの書き込みまたは読み取りに使用されます。[**クライアントプリフィックスの共有**] チェックボックスをオンにすると、Integration Server が Universal Messaging 領域間で同じログキューから読み書きできるようにするクライアントプリフィックスが Universal Messaging キュー名に含まれます。ログキューとして Universal Messaging キューを使用する方法の詳細については、『*webMethods Audit Logging Guide*』を参照してください。

5. [**接続設定**] で、以下を指定します。

フィールド	説明
[領域 URL]	Universal Messaging サーバの URL。形式は *protocol*:// *UM_host*: *UM_port* で (*protocol* は nsp、nsps、nhps などのポートのプロトコル)、`nsp://127.0.0.1:9000`、`nsps://localhost:9000` などのように指定します。 Universal Messaging サーバに接続するプロトコルが nsps または nhps の場合、接続エイリアスに使用する適切な認証を指定する必要があります。トラストストアエイリアス、またはトラストストアエイリアスとキーストアエイリアスを指定する必要があります。 サーバのクラスタを使用する場合は、カンマ区切りリストまたはセミコロン区切りリストを使用して、クラスタ内の各サーバの URL リストを入力します。デフォルトで、Integration Server はクラスタ内のマスター領域サーバに従うよう設定されています。これは、Integration Server がクラスタ内のマスター領域サーバに常に接続されることを示

フィールド	説明
	します。Software AG は、エイリアスがマスター追跡を使用する場合、URL のカンマ区切りリストを使用することをお勧めします。 **注:** クラスタ内の各 Universal Messagingサーバの URL を指定する必要があります。Integration Server は、URL が指定されている Universal Messaging サーバのみに接続します。その後、Integration Server は、指定された Universal Messaging サーバでのみチャネルと継続的サブスクリプションなどのリソースを作成します。クラスタで Universal Messaging サーバを 1 台だけ指定すると、リソースはサーバのみに対してローカルになります。 このメッセージング接続エイリアスと共に Universal Messaging の水平スケーラビリティ機能を使用するには、それぞれの領域 URL またはカンマで区切られた複数の領域 URL をカッコで囲みます。クラスタ内で、領域 URL はカンマで区切る必要があります。 **水平スケーラビリティの URL 構文の例** ■ (*realm1*)(*realm2*)(*realm3*)(*realm4*): 4 つのスタンドアロン領域 *realm1*、*realm2*、*realm3*、*realm4* を示します。 ■ (*realm1*,*realm2*)(*realm3*,*realm4*): 2 つのクラスタを示します。一方は *realm1* と *realm2* から構成され、もう一方は *realm3* と *realm4* から構成されます。 ■ (*realm1*)(*realm2*,*realm3*)(*realm4*): *realm2* と *realm3* から構成される 1 つのクラスタ、および *realm1* と *realm4* という名前の 2 つのスタンドアロン領域を示します。 このエイリアスを使用するプロデューサやコンシューマのマスター追跡を無効にした場合、クラスタの領域サーバを区切るときにカンマを使用するかセミコロンを使用するかによって、Integration Server の接続先の Universal Messaging 領域サーバが決まります。 ■ Integration Server が常にリストの最初の Universal Messaging サーバに接続を試みる場合は、カンマを使用して URL を区切ります。リストの 2 番目の Universal Messaging サーバに接続を試みるのは、最初のサーバが使用不可になった場合だけです。 ■ Integration Server がリストからランダムに選択された URL に接続する場合は、セミコロンを使用して URL を区切ります。この場合は、クライアントがクラスタ内のサーバに適度に分散します。 無効化の方法を含め、マスター追跡動作の詳細については、273 ページの「webMethods メッセージングのマスター追跡について」を参照してください。

フィールド	説明
[最大再接続試行数]	Universal Messaging への接続が失敗した場合に、Integration Server が行う再接続の最大試行回数を指定します。接続を再確立できない場合、Integration Server はメッセージをエラーログに書き込み、このメッセージング接続エイリアスによって作成された接続は停止します。デフォルトの試行回数は 5 回です。 **注:** [**最大再接続試行数**] の値は、既存の接続が失敗した場合のみ考慮されます。[**最大再接続試行数**] の値は、Integration Server がメッセージング接続エイリアスの起動時に考慮する要因ではありません。

6. [**プロデューサの設定**] で、以下を指定します。

フィールド	説明
[CSQ の有効化]	クライアントサイドキューがこの Universal Messaging 接続エイリアスと共に使用されるかどうか。次のいずれかの手順に従います。 ■ このクライアントサイドキューを Universal Messaging 接続エイリアスと共に使用する場合は、[**CSQ の有効化**] チェックボックスをオンにします。クライアントサイドキューが使用されているときに、Integration Server がこの Universal Messaging 接続エイリアスを使用してドキュメントをパブリッシュするときに Universal Messaging サーバが使用可能でない場合、Integration Server はドキュメントをクライアントサイドキューに書き込みます。 ■ このクライアントサイドキューを Universal Messaging 接続エイリアスと共に使用しない場合は、[**CSQ の有効化**] チェックボックスをオフにします。クライアントサイドキューが使用されていないとき、Integration Server がこの Universal Messaging 接続エイリアスを使用してドキュメントをパブリッシュするときに Universal Messaging サーバが使用可能でない場合、パブリッシュサービスは ISRuntimeException で終了します。
[最大 CSQ サイズ (メッセージ)]	この Universal Messaging 接続エイリアスのクライアントサイドキューに入れることができるドキュメント (メッセージ) の最大数。クライアントサイドキューが最大容量に達すると、この接続エイリアスを使用するパブリッシュサービスは ISRuntimeException で終了します。

フィールド	説明
	クライアントサイドキューで無制限の数のメッセージを格納できるようにする場合は、「-1」を指定します。
[CSQ を順番に排出]	Integration Server がこのエイリアスのクライアントサイドキューを排出する際に、Integration Server がクライアントサイドキューにメッセージを入れた順番で Universal Messaging サーバにメッセージを送信するかどうか。次のいずれかの手順に従います。 ■ Universal Messaging サーバへの接続が再確立されたときに、Integration Server がパブリケーション順序でメッセージを Universal Messaging に送信する場合、[**CSQ を順番に排出**] チェックボックスをオンにします。Universal Messaging サーバへの接続が再確立された後、クライアントサイドキューが完全に排出されるまで、Integration Server は新規にパブリッシュされたメッセージをクライアントサイドキューに書き込み続けます。 ■ Universal Messaging サーバへの接続が再確立されたときに、Integration Server でパブリケーション順序を保持しない場合、[**CSQ を順番に排出**] チェックボックスをオフにします。Universal Messaging サーバへの接続が再確立された後、Integration Server はクライアントサイドキューを排出しながら、新規にパブリッシュされたメッセージを Universal Messaging サーバに直接送信します。
[再接続中のパブリッシュ待機時間]	この Universal Messaging 接続エイリアスを使用するパブリッシュサービスが、Universal Messaging サーバへの接続が失敗後に再確立されるまで待機するミリ秒数。[**再接続中のパブリッシュ待機時間**] が経過する前に Integration Server が接続を再確立すると、パブリッシュサービスは実行を継続します。指定した時間が経過しても接続が再確立されない場合、パブリッシュサービスは ISRuntimeException で終了します。デフォルトは 0 ミリ秒で、パブリッシュサービスは Integration Server による接続の再確立を待機しません。 **注:** Universal Messaging 接続エイリアスが Universal Messaging クラスタに接続するように設定されている場合は、いずれかのメンバーサーバの障害後に Universal Messaging クラスタがクォーラムを確立するまでの時間が収まる程度に [**再接続中のパブ**

フィールド	説明
	リッシュ待機時間] の値が十分長いことを確認してください。
[プロデューサに対してマスター追跡を有効にする]	Integration Server がこの Universal Messaging 接続エイリアスを使用してメッセージをパブリッシュするときに、Integration Server は Universal Messaging クラスタのマスター領域サーバに常に接続するかどうか。次のいずれかの手順に従います。 ■ Integration Server がこの Universal Messaging 接続エイリアスを使用してメッセージをパブリッシュするときに Integration Server が常に Universal Messaging クラスタのマスター領域サーバに接続することを示すには、[**プロデューサに対してマスター追跡を有効にする**] チェックボックスをオンにします。 ■ この接続エイリアスを使用してメッセージを送信するときにエイリアスのマスター追跡動作を無効にするには、[**プロデューサに対してマスター追跡を有効にする**] チェックボックスをオフにします。 プロデューサに対してマスター追跡を無効にした場合、Integration Server は [**領域 URL**] パラメータの URL のカンマまたはセミコロン区切りリストで指定された順序で Universal Messaging クラスタのサーバに接続します。 マスター追跡動作の詳細については、273 ページの「webMethods メッセージングのマスター追跡について」を参照してください。

7. [**コンシューマ設定**] で、[**要求応答チャネルおよびリスナーを有効にする**] チェックボックスを以下のように指定します。

フィールド	説明
[要求応答チャネルおよびリスナーを有効にする]	この Universal Messaging 接続エイリアスを要求/応答または「パブリッシュして待機」シナリオの一部として使用し、要求ドキュメントを送信したり、応答ドキュメントを送信したり、応答ドキュメントを受信したりするかどうか。次のいずれかの手順に従います。 ■ この Universal Messaging 接続エイリアスを要求/応答または「パブリッシュして待機」シナリオの一部として使用し、要求ドキュメントを送信したり、応答ドキュメントを送信したり、応答ドキュ

フィールド	説明
	メントを受信したりする場合は、[**要求応答チャネルおよびリスナーを有効にする**] チェックボックスをオンにします。 ■ この Universal Messaging 接続エイリアスを要求/応答または「パブリッシュして待機」シナリオの一部として使用しない場合は、[**要求応答チャネルおよびリスナーを有効にする**] チェックボックスをオフにします。これによって、Integration Server と Universal Messaging のリソースが節約されます。 [**要求応答チャネルおよびリスナーを有効にする**] チェックボックスをオンにすると、Universal Messaging 接続エイリアスが起動した際に、Integration Server は以下のことを実行します。 ■ この Universal Messaging 接続エイリアスの要求/応答チャネルがまだない場合は、要求/応答チャネルを作成します。要求/応答チャネル名には、Integration Server を識別するストリング、クライアントプリフィックス、および「RequestReply」というテキストが含まれます。 ■ Integration Server で、このエイリアス固有の要求/応答チャネルにサブスクライブするリスナーを起動します。
[コンシューマに対してマスター追跡を有効にする]	Integration Server がこの Universal Messaging 接続エイリアスを使用してメッセージを抽出するときに、Integration Server は Universal Messaging クラスタのマスター領域サーバに常に接続するかどうか。次のいずれかの手順に従います。 ■ この Universal Messaging 接続エイリアスを使用してメッセージを抽出するときに Integration Server が常に Universal Messaging クラスタのマスター領域サーバに接続することを示すには、[**コンシューマに対してマスター追跡を有効にする**] チェックボックスをオンにします。 ■ この接続エイリアスを使用してメッセージを抽出するときにエイリアスのマスター追跡動作を無効にするには、[**コンシューマに対してマスター追跡を有効にする**] チェックボックスをオフにします。 このエイリアスを使用するコンシューマに対するマスター追跡を無効にした場合、Integration Server は [**領域 URL**] パラメータの URL のカンマまたはセミコロン

フィールド	説明
	区切りリストで指定された順序で Universal Messaging クラスタのサーバに接続します。 マスター追跡動作の詳細については、273 ページの「webMethods メッセージングのマスター追跡について」を参照してください。

8. [**クライアント認証の設定**] で、以下を指定します。

フィールド	説明
[クライアント認証]	Integration Server クライアントが Universal Messaging サーバに接続するために使用する認証のタイプ。以下のいずれかのオプションを選択します。 ■ [**なし**]認証が行われないようにする場合は、このオプションを選択します。これがデフォルトです。 Universal Messaging がクライアント認証を行わない場合でも、[**領域 URL**] に指定されたプロトコルが nsps または nhps である場合、[**トラストストアエイリアス**] フィールドでトラストストアエイリアスを選択する必要があります。 ■ [**ユーザ名/パスワード**]Universal Messaging サーバでユーザ名とパスワードの組み合わせを使用した基本クライアント認証を実行する場合は、このオプションを選択します。このオプションを選択した場合は、[**ユーザ名**] および [**パスワード**] フィールドでクライアントが使用するユーザ名およびパスワードを指定します。 クライアント認証にユーザ名とパスワードを指定することに加え、[**領域 URL**] に指定されたプロトコルが nsps または nhps である場合、[**トラストストアエイリアス**] フィールドでトラストストアエイリアスを選択する必要もあります。 ■ [**認証ベース**]Integration Server が Universal Messaging の SSL (Secure Sockets Layer) ポートに接続し、Universal Messaging が認証ベースの認証を行う場合、このオプションを選択します。 Universal Messaging ポートの設定によっては、Universal Messaging 接続エイリアスに対してトラストストアエイリアスと、場合によってはキーストアエイリアスとキーエイリアスを指定する必要があります。 ■ Universal Messaging ポートに [**Enable Client Cert Validation**] オプションがオンになっていない場合、トラストストアエイリアスを指定する必要があります。 ■ Universal Messaging ポートに [**Enable Client Cert Validation**] オプションがオンになっている場合、トラストストアエイリア

フィールド	説明
	ス、キーストアエイリアス、およびキーエイリアスを指定する必要があります。 **注:** ポートに対して指定されたプロトコルおよび [**Enable Client Cert Validation**] オプションを表示するには、Universal Messaging Enterprise Manager を使用します。
[**トラストストアエイリアス**]	Universal Messaging サーバの認証用の認証局 (CA) を含むトラストストアのエイリアスを作成します。 Universal Messaging ポートが証明書ベースの認証用に設定されている場合、または [**領域 URL**] に指定されたプロトコルが nsps または nhps である場合、トラストストアエイリアスを選択する必要があります。
[**キーストアエイリアス**]	Universal Messaging ポートへの接続時に Integration Server で使用するクライアント認証が含まれるキーストアのエイリアス。Universal Messaging ポートが証明書ベースの認証用に設定されている場合は、キーストアエイリアスを選択する必要があります。
[**キーエイリアス**]	Universal Messaging ポートに安全に接続するプライベートキーを含むキーのエイリアス。このキーエイリアスは、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。 [**キーストアエイリアス**] を指定した場合は、[**キーエイリアス**] を指定する必要があります。

9. [**拡張ロギング**] で以下の内容を指定し、この接続エイリアスを使用して送受信されるメッセージに関する追加ログを設定します。

フィールド	説明
[**ログタイプ**]	このメッセージング接続エイリアスを使用してドキュメントを送受信するメッセージプロデューサやコンシューマに対し拡張ロギングが有効になっている場合に、Integration Server がログメッセージを書き込む先を指定します。 ■ [**サーバログ**]。拡張ロギングメッセージをサーバログに書き込む場合は、このオプションを選択します。サーバログを宛先に指定する場合、0168 メッセージング (拡張ロギング) サーバログ機能のログレベルを Info 以上に上げてください。 ■ [**メッセージング監査ログ**]。拡張ロギングメッセージをメッセージング監査ログに書き込みます。

フィールド	説明
	上記のいずれかのオプションのみを選択できます。Integration Server は拡張ロギングメッセージをサーバログとメッセージング監査ログの両方に書き込むことはできません。
[プロデューサメッセージ ID の追跡を有効化]	選択した場合、メッセージプロデューサがこのメッセージング接続エイリアスを使用して **[プロデューサメッセージ ID 追跡: インクルードされたチャネル]** に示されたパブリッシュ可能なドキュメントタイプのインスタンスであるドキュメントをパブリッシュするとき、Integration Server が追加ログメッセージを書き込むことを示します。
[プロデューサメッセージID追跡:インクルードされたチャネル]	Integration Server がパブリッシュ時に追加ログを記録する対象のパブリッシュ可能ドキュメントタイプに対応する各チャネル名。セミコロン (;) を使用して、各チャネル名を区切ります。このメッセージング接続エイリアスがメッセージを送信するすべてのチャネルで Integration Server による拡張ロギングを実行したい場合、このフィールドを空白のままにします。
[コンシューマメッセージ ID の追跡を有効化]	選択した場合、Integration Server が、このメッセージング接続エイリアスを使用してメッセージを受信するメッセージングコンシューマ (トリガー) の追加ログメッセージを書き込むことを示します。Integration Server は **[コンシューマメッセージ ID 追跡: インクルードされたトリガー]** に示されたトリガーの追加ログメッセージを書き込みます。
[コンシューマメッセージID追跡:インクルードされたトリガー]	Integration Server がトリガー処理時に追加ログを記録する対象のトリガーの完全修飾名。セミコロン (;) を使用して、各トリガー名を区切ります。このメッセージング接続エイリアスを使用してメッセージを受信するすべてのトリガーで Integration Server による拡張ロギングを実行したい場合、このフィールドを空白のままにします。

Universal Messaging と送受信されるドキュメントの拡張ロギングの使用方法については、357 ページの「メッセージングの拡張ロギングの使用」を参照してください。

10. **[変更内容の保存]** をクリックします。

11. Universal Messaging 接続エイリアスを有効にします。

注:
証明書ベースの認証を行う Universal Messaging ポートに接続するよう Universal Messaging 接続エイリアスを設定するには、エイリアスはトラストストアエイリアス、キーストアエイリアス、およびキーエイリアスを指定する必要があります。ただし、接続エイリアスで **[クライアントの認証]** を **[証明書ベース]** に設定しても、Universal Messaging サーバとの SSL 接続を確立するために必要なトラストストアエイリアスまたはキーストアエイリアスを指定しなければ、Integration Server は JVM でこの情報を検索します。JVM で SSL 関連のシステムプロパティを設定する詳細については、498 ページの「Integration Server JVM の SSL 情報の設定」を参照してください。

Software AG は、JVM システムに依存せずに、トラストストアエイリアス、キーストアエイリアス、キーエイリアス情報を Universal Messaging 接続エイリアスで指定することをお勧めします。

webMethods メッセージングのマスター追跡について

マスター追跡は、Universal Messaging クライアント設定の 1 つであり、クライアントセッションは常に Universal Messaging クラスタのマスタ領域サーバに接続されていることを示します。マスター領域サーバが使用できない場合、Integration Server は接続を確立しません。Universal Messaging 接続エイリアスの場合、エイリアスから作成されたクライアントセッションがプロデューサ接続またはコンシューマ接続のどちらか一方または両方に対してマスター追跡をするか、またはどちらもしないかを制御できます。

多くの場合、マスター追跡を行うと、パフォーマンスを向上させることができます。デフォルトで、各 Universal Messaging 接続エイリアスは、プロデューサ接続とコンシューマ接続に対してマスター追跡するよう設定されます。Universal Messaging 接続エイリアスを使用してメッセージをパブリッシュするサービス用に作成された接続はマスター領域サーバを追跡します。Universal Messaging 接続エイリアスを使用してメッセージを受信する webMethods messaging trigger はマスター領域サーバを追跡する接続を使用します。

ただし、プロデューサ接続やコンシューマ接続がマスター領域サーバを追跡しないようにする場合もあります。たとえば、ソリューションに多くの接続を使用する多数の webMethods messaging trigger がある場合、Universal Messaging クラスタ間で負荷を分散した方がパフォーマンスが向上することがあります。この場合、Universal Messaging 接続エイリアスを使用してコンシューマに対してマスター追跡を無効にします。

Universal Messaging 接続エイリアスの場合、Integration Server は、エイリアスから作成されたクライアントセッションがマスター追跡するかどうかを示すプロデューサオプションとコンシューマオプションを提供します。

- メッセージプロデューサの場合、[**プロデューサに対してマスター追跡を有効にする**] チェックボックスによって、メッセージプロデューサに対して確立されたクライアントセッションが常にマスター領域サーバに接続されるかどうかが決まります。
- メッセージコンシューマの場合、[**コンシューマに対してマスター追跡を有効にする**] チェックボックスによって、webMethods messaging trigger など、メッセージコンシューマに対して確立されたクライアントセッションが常にマスター領域サーバに接続されるかどうかが決まります。

Universal Messaging 接続エイリアスを使用するプロデューサやコンシューマ対してマスター追跡動作が無効になっている場合、[**領域 URL**] パラメータに表示される領域サーバを区切るときにカンマを使用するかセミコロンを使用するかによって、Integration Server の接続先の Universal Messaging 領域サーバが決まります。[**領域 URL**] パラメータの詳細については、262 ページの「Universal Messaging 接続エイリアスの作成」を参照してください。

注:

- Integration Server 10.0 より前のバージョンで、webMethods メッセージングのマスター追跡動作を無効にする唯一の方法は custom_wrapper.conf ファイルを変更して `wrapper.java.additional.n=-DFollowTheMaster=false` を含めることでした。ここで `n` は次に使用可能な wrapper.java.additional 番号です。`DFollowTheMaster` パラメータを使用

して Integration Server のマスター追跡動作を制御しないでください。このパラメータ設定は、Universal Messaging 接続エイリアスおよび JMS 接続エイリアスに設定されたマスター追跡動作を上書きします。

- webMethods Deployer で `Delete Universal Messaging Durable` プロパティが [**Yes**] に設定されている場合、Integration Server は Universal Messaging から継続性を削除します。このプロパティおよび Deployer の使用方法の詳細については、*webMethods Deployer User's Guide*を参照してください。

Universal Messaging を JMS プロバイダとして使用する JMS 接続エイリアスのマスター追跡の設定の詳細については、307 ページの「JMS 接続エイリアスの作成」の [**マスター追跡を有効にする**] オプションの説明を参照してください。

メッセージング接続エイリアスの編集

自分で作成したメッセージング接続エイリアスや Integration Server によって作成された一部のデフォルトのメッセージング接続エイリアスについて、プロパティを編集することができます。メッセージング接続エイリアスを編集する前に、以下の点を考慮してください。

- 接続エイリアス名を除く、既存のメッセージング接続エイリアスのすべてのプロパティを編集できます。
- メッセージング接続エイリアスは、編集する前に無効にする必要があります。エイリアスが無効になっていない場合、*messagingConnectionAlias* 名の編集リンクはハイパーリンクされません。
- Universal Messaging の既存の接続エイリアスのクライアントプリフィックスを変更した場合、Integration Server はエイリアスを使用するトリガー用の新しい継続的オブジェクトを Universal Messaging に作成します。古いクライアントプリフィックスを使用する継続的オブジェクトは Universal Messaging から自動的には削除されません。古い継続的オブジェクトはメッセージを含んでいることや、メッセージの受信を続行することがあります。
- IS_DES_CONNECTION エイリアスの場合、説明とクライアントプリフィックスのみ編集できます。Integration Server がクラスタ (ステートレスまたはステートフル) の一部である場合、IS_DES_CONNECTION 接続エイリアスのクライアントプリフィックスは各 Integration Server で同じである必要があります。
- IS_LOCAL_CONNECTION エイリアスは編集できません。

メッセージング接続エイリアスを編集するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods の設定] に移動します。
3. [**webMethods メッセージング接続エイリアスの定義**] テーブルで、編集するメッセージング接続エイリアスの名前をクリックします。
4. 編集しているのが Broker 接続エイリアス、Universal Messaging、または Digital Event Services 接続エイリアスであるかに応じて、以下のいずれかを行います。

- [**Broker メッセージング接続エイリアスの編集**] をクリックします。
- [**Universal Messaging 接続エイリアスの編集**] をクリックします。
- [**Digital Event Services接続エイリアスの編集**] をクリックします。

注:
[***messagingConnectionAliasType* の編集**] リンクが表示されるが使用できない場合、つまり、ハイパーリンクされていない場合、メッセージング接続エイリアスが無効になっていないことがあります。無効になったメッセージング接続エイリアスのみ編集できます。

5. メッセージング接続エイリアスの情報を更新します。

6. [**変更内容の保存**] をクリックします。

7. メッセージング接続エイリアスを有効にします。メッセージング接続エイリアスの有効化の詳細については、276 ページの「メッセージング接続エイリアスの有効化」を参照してください。

8. これが Broker 接続エイリアスの場合、変更を適用するには、Integration Server を再起動する必要があります。

9. これが、Universal Messaging キューへのログエントリの書き込みまたは読み取りに使用される Universal Messaging 接続エイリアスである場合、変更した内容を有効にするには、Integration Server を再起動する必要があります。

メモ:

- Broker 接続エイリアスを別のテリトリにある Broker に切り替えると、切り替え先の Broker で、パブリッシュ可能なドキュメントタイプの同期が必要な場合があります。パブリッシュ可能なドキュメントタイプの同期については、*webMethods Service Development Help*を参照してください。
- [**要求応答チャネルおよびリスナーを有効にする**] チェックボックスをオンにしている Universal Messaging 接続エイリアスのクライアントプリフィックスを変更すると、Universal Messaging は新しいクライアントプリフィックスの付いた新しい要求/応答チャネルを作成します。ただし、Universal Messaging は、Universal Messaging 接続エイリアスの以前の要求/応答チャネルを削除しません。Universal Messaging Enterprise Manager を使用して、古いチャネルを削除してください。
- [**要求応答チャネルおよびリスナーを有効にする**] チェックボックスをオフに変更して Universal Messaging 接続アイリアスを編集した場合、Universal Messaging 接続エイリアスの要求/応答チャネルは Universal Messaging サーバに保持されます。チャネルを削除するには、Universal Messaging Enterprise Manager を使用してください。
- ロガーがログエントリを Universal Messaging キューに書き込むかログエントリを読み取るときに使用する Universal Messaging 接続エイリアスの場合、[**クライアントプリフィックスの共有**] チェックボックスの状態によって、エイリアスのクライアントプリフィックスが Universal Messaging キュー名に含まれるかどうかが決まります。[**クライアントプリフィックスの共有**] チェックボックスをオンまたはオフにすることにより Universal Messaging 接続エイリアスを編集し、そのエイリアスがロガーによって使用される場合、キュー名の変更を

有効にするには、Integration Server を再起動する必要があります。また、そのエイリアスを使用する Integration Server のクラスタまたは非クラスタグループ内の他の Integration Server に同じ変更を行う必要があります。

メッセージング接続エイリアスの有効化

メッセージング接続エイリアスを有効にすると、Integration Server でメッセージング接続エイリアスを使用して、パブリッシュサービスおよび webMethods messaging trigger の代わりにメッセージングプロバイダとの間でメッセージの送信および受信を実行できるようになります。メッセージング接続エイリアスに関しては、次の情報に注意してください。

- Universal Messaging 接続エイリアスを有効にすると、変更はただちに有効になります。パブリッシュサービスおよびトリガーは、メッセージングプロバイダとの間でメッセージを送信および受信するために、エイリアスを使用し始めます。
- Broker 接続エイリアスを有効にすると、変更を有効にするために Integration Server を再起動する必要があります。パブリッシュサービスおよび webMethods messaging triggerは、Integration Server が再起動するまで、ドキュメントを送信および受信するために、エイリアスを使用し始めません。
- IS_LOCAL_CONNECTION メッセージング接続エイリアスは、有効/無効を切り替えられません。エイリアスは、ローカルパブリッシュのために Integration Server で常に使用できます。

メッセージング接続エイリアスを有効にするには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods の設定] に移動します。
3. [**webMethods メッセージング接続エイリアスの定義**] リストで、有効にするメッセージング接続エイリアスを探します。
4. [**有効**] 列の [**いいえ**] をクリックして、メッセージング接続エイリアスを有効にします。

   Universal Messaging 接続エイリアスの場合、Integration Server Administrator は [**いいえ**] を ✓ [**はい**] に置き換えます。

   Broker 接続エイリアスの場合、Integration Server Administrator は [**いいえ**] を ✓ [**はい (再開を待機中)**] に置き換えます。
5. これが Broker 接続エイリアスの場合、変更を適用するには、Integration Server を再起動してください。

メッセージ接続エイリアスの無効化について

Integration Server では、任意のメッセージング接続エイリアスを無効にすることができます。無効化されたメッセージング接続エイリアスは、エイリアスで指定されたメッセージングプロバ

イダとの間でメッセージを送信および受信するために使用できなくなります。具体的には、メッセージング接続エイリアスが無効になると、以下のようになります。

- 無効化されたエイリアスを使用するパブリッシュサービスは、ISRuntimeException で終了します。
- エイリアスを使用する webMethods messaging trigger は、無効になります。

Universal Messaging 接続エイリアスを無効にしたときは、変更を有効にするために Integration Server を再起動する必要はありません。

- エイリアスを使用するパブリッシュサービスは、メッセージング接続エイリアスが無効になるとただちに ISRuntimeException を受信します。
- エイリアスを使用する webMethods messaging trigger を一時停止する前に、Integration Server は、トリガーが既に受信したメッセージの処理が終わるまでトリガーを短時間待機します。Integration Server は、(成功、失敗にかかわらず) トリガーサービスの実行が完了し、トリガーがメッセージプロバイダへのメッセージの受信を確認すると、メッセージの処理が完了したと判断します。トリガーサービスの稼働時間が長く、割り当てられた時間内に完了できない場合、Integration Server はトリガーを無効にし、メッセージの処理が未完了であると判断します。

  注:
  トリガーを一時停止しても、トリガーサービスは停止しません。このため、Integration Server が webMethods messaging trigger を一時停止した後でも、トリガーサービスは完了するまで実行されます。トリガーサービスが完了すると、Integration Server はメッセージングプロバイダへのメッセージの受信を確認しようとします。しかし、メッセージング接続エイリアスが無効であるため、この受信確認は失敗します。メッセージが保証付きの場合、メッセージング接続エイリアスが有効になりトリガーが再開されると、メッセージングプロバイダはメッセージを再配信します。メッセージの再配信によって、重複処理が発生することがあります。

Broker 接続エイリアスを無効にすると、エイリアスに対する変更を有効にするために Integration Server を再起動する必要があります。Integration Server が再起動するまで、Broker 接続エイリアスは実際には無効になりません。つまり、エイリアスを使用するパブリッシュサービスおよび webMethods messaging triggerは、再起動が発生するまでメッセージの送信および受信を続けます。再起動すると、Integration Server はメッセージング接続エイリアスを使用する webMethods messaging trigger を一時停止します。また、再起動後、メッセージング接続エイリアスを使用するパブリッシュサービスは ISRuntimeException で終了します。

注:
IS_LOCAL_CONNECTION メッセージング接続エイリアスは、有効/無効を切り替えられません。エイリアスは、ローカルパブリッシュのために Integration Server で常に使用できます。

メッセージング接続エイリアスの無効化

以下の場合に、Universal Messaging 接続エイリアスを無効化することがあります。

- メッセージング接続エイリアスを編集する。
- サービスがメッセージングプロバイダにメッセージをパブリッシュしないようにする。

- 特定のメッセージングプロバイダからのメッセージ受信を停止する。
- 特定のメッセージングプロバイダを使用するすべての webMethods messaging triggerを停止する。

Universal Messaging 接続エイリアスは Integration Server の再起動を必要とせずに変更が有効になるため、エイリアスを無効にすることで、エイリアスにおける Universal Messaging サーバからのメッセージの送信、抽出、処理を簡単に停止することができます。

メッセージング接続エイリアスを無効にするには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods の設定] に移動します。
3. [**webMethods メッセージング接続エイリアスの定義**] リストで、無効にするメッセージング接続エイリアスを探します。
4. [**有効**] 列にある ✓ アイコンをクリックして、メッセージング接続エイリアスを無効にします。

   Universal Messaging 接続エイリアスの場合、Integration Server Administrator は ✓ を [**いいえ**] に置き換えます。

   Broker 接続エイリアスの場合、Integration Server Administrator は ✓ [**はい**] を ✓ [**いいえ (再開を待機中)**] に置き換えます。
5. これが Broker 接続エイリアスの場合、変更を適用するには、Integration Server を再起動してください。

メッセージング接続エイリアスの状態

メッセージング接続エイリアスの [**有効**] 列には、以下のいずれかの状態が表示されます。

[有効] 列の状態	説明
✓ [**はい**]	メッセージング接続エイリアスは有効です。Integration Server は、メッセージングプロバイダに接続していて、メッセージを送受信するためにエイリアスを使用することができます。
✓ [**はい(再開を待機中)**]	メッセージング接続エイリアスは現在無効ですが、Integration Server の再起動後に有効になります。Integration Server は、メッセージングプロバイダに接続されていません。Integration Server は、メッセージを送受信するためにエイリアスを使用することができません。
✓ [**はい(未接続)**]	メッセージング接続エイリアスは有効ですが、Integration Server はメッセージングプロバイダに接続されていません。

[有効] 列の状態	説明
	Integration Server は、メッセージを送受信するためにエイリアスを使用することができません。
[いいえ]	メッセージング接続エイリアスは有効ではありません。Integration Server は、メッセージを送受信するためにエイリアスを使用することができません。
✓ **[いいえ(再開を待機中)]**	メッセージング接続エイリアスは現在有効ですが、Integration Server の再起動後に無効になります。現在、Integration Server は Broker に接続されています。サービスはメッセージを Broker にパブリッシュすることができ、トリガーは Broker からメッセージを抽出することができます。

注:
Integration Server Administrator では、事前定義済みの IS_LOCAL_CONNECTION メッセージング接続エイリアスのステータスは表示されません。エイリアスは、有効/無効を切り替えられず、ローカルパブリッシュのために Integration Server で常に使用できます。

デフォルトのメッセージング接続エイリアスの指定

デフォルトのメッセージング接続エイリアスにより、パブリッシュ可能なドキュメントタイプの [**接続エイリアス名**] プロパティがデフォルトに設定されているか空の場合にメッセージの送信先となるメッセージングプロバイダが決まります。

- パブリッシュ可能なドキュメントタイプのインスタンスをパブリッシュするサービスは、デフォルトのメッセージング接続エイリアスを使用して、メッセージングプロバイダに接続し、ドキュメントをパブリッシュします。
- パブリッシュ可能なドキュメントタイプにサブスクライブする webMethods messaging trigger は、デフォルトのメッセージング接続エイリアスを使用して、プロバイダに接続し、メッセージを抽出します。

つまり、パブリッシュ可能なドキュメントタイプで [**接続エイリアス名**] がデフォルトに設定されているか空の場合は、デフォルトのメッセージング接続エイリアスによって、パブリッシュされるドキュメントインスタンスを受信しルーティングするメッセージングプロバイダが決まります。

はじめて Integration Server を起動すると、Integration Server は Integration Server が以前のバージョンから移行されたかどうか、および同じ場所にインストールされている他の製品に応じて、デフォルトのメッセージング接続エイリアスを設定します。

デフォルトのメッセージング接続エイリアスを変更するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods の設定] に移動します。

3. [**デフォルト接続エイリアスの変更**] をクリックします。

4. [**新しいデフォルトのエイリアス接続の選択**] の [**接続エイリアス名**] リストで、新しいデフォルトのメッセージング接続エイリアスとして使用するメッセージング接続エイリアス名を選択します。

5. [**更新**] をクリックします。

メモ

- デフォルトの接続エイリアスを変更すると、Integration Server は、デフォルトのメッセージング接続エイリアスを使用するパブリッシュ可能なドキュメントタイプにサブスクライブするすべての webMethods messaging triggerを再ロードします。
- デフォルトの接続エイリアスを変更したら、デフォルトのエイリアスを使用するパブリッシュ可能なドキュメントタイプをメッセージングプロバイダと同期する必要があります。これにより、新しいデフォルトのメッセージングプロバイダに関するプロバイダの定義は、デフォルトのメッセージング接続エイリアスを使用するパブリッシュ可能なドキュメントタイプと同期されます。

メッセージング接続エイリアスの削除

メッセージング接続エイリアスを削除する前に、次の点に注意してください。

- メッセージング接続エイリアスは、削除前に無効にする必要があります。メッセージング接続エイリアスの無効化の詳細については、277 ページの「メッセージング接続エイリアスの無効化」を参照してください。
- メッセージング接続エイリアスは、削除時にパブリッシュ可能なドキュメントタイプに割り当てないでください。Integration Server は、パブリッシュ可能なドキュメントタイプによって使用されるメッセージング接続エイリアスが削除されることを防止できません。
- 事前定義済みのメッセージング接続エイリアス IS_LOCAL_CONNECTION は削除できません。
- 事前定義済みのメッセージング接続エイリアス IS_DES_CONNECTION は削除できません。

メッセージング接続エイリアスを削除するには

1. Integration Server Administrator を開きます。

2. [**メッセージング**] > [webMethods の設定] に移動します。

3. [**webMethods メッセージング接続エイリアスの定義**] テーブルで、メッセージング接続エイリアスが無効になっていない場合は無効にします。

4. [**削除**] 列で、削除するメッセージング接続エイリアスの × をクリックします。

5. [**OK**] をクリックして、メッセージング接続エイリアスの削除を確認します。

Universal Messaging Server への接続の認証

Software AG Universal Messaging がメッセージングプロバイダとして機能するときは、Integration Server はクライアントとして機能し、Universal Messaging 領域サーバはサーバとして機能します。Integration Server と Universal Messaging サーバとの間の通信は、Universal Messaging サーバ上の ACL 管理を利用してセキュリティ保護されます。Universal Messaging サーバが ACL を適用する前段階でユーザ名とパスワードを使用して接続を認証することにより、Integration Server と Universal Messaging との間の接続のセキュリティをさらに強化することができます。この認証方法は、Universal Messaging 接続エイリアスを作成するときに指定します。

注:
この機能は Integration Server 9.8 以降および Universal Messaging 9.6 以降に適用されます。

Integration Server と Universal Messaging との間の接続をこの方法で認証するように Universal Messaging 接続エイリアスを設定するときは、SASL (Simple Authentication and Security Layer) フレームワークまたは JAAS (Java Authentication and Authorization Service) フレームワークを使用してユーザクレデンシャルが交換されます。Universal Messaging では、デフォルトで JAAS フレームワークを使用します。

使用するフレームワークは Universal Messaging 管理者か決定します。接続をテストする前に、Universal Messaging 管理者と協力して以下の項目を用意します。

- **SASL フレームワーク** このフレームワークでは、Universal Messaging サーバは指定されたディレクトリインスタンスに基づいてユーザクレデンシャルを検証します。クレデンシャルは、内部ディレクトリまたは外部ディレクトリ (LDAP など) に格納されます。Universal Messaging 管理者と協力して以下の項目を用意します。
  - **内部ユーザリポジトリ**ユーザクレデンシャルを内部ユーザリポジトリに格納する場合、Universal Messaging 管理者は Software AG Security Infrastructure コマンドラインツールを使用してリポジトリを作成する必要があります。また、Universal Messaging 管理者は Nirvana.directory.provider システムプロパティを後述するサーバプロパティ設定表に示すとおりに設定する必要もあります。
  - **外部ユーザリポジトリ設定**ユーザクレデンシャルを LDAP などの外部リポジトリに格納する場合、Universal Messaging 管理者は Nirvana.directory.provider システムプロパティを後述するサーバプロパティ設定表に示すとおりに設定する必要があります。
  - **サーバプロパティ設定**Universal Messaging 管理者は、nserver.conf ファイル (Universal Messaging サーバの *Universal Messaging_directory* \server\umserver\bin\ にあります) でプロパティを以下のように設定する必要があります。

設定するプロパティ	設定値
Nirvana.auth.sagrepo.path	Security Infrastructure コマンドラインツールによって作成されたユーザクレデンシャルテキストファイルの相対パスまたは絶対パス。
Nirvana.directory.provider	以下のいずれか。

設定するプロパティ	設定値

	■ ユーザクレデンシャルが内部ユーザリポジトリに格納される場合は、このプロパティを
	
	`com.pcbsys.foundation.security.auth. fSAGInternalUserRepositoryAd`
	
	■ ユーザクレデンシャルが外部ユーザリポジトリに格納される場合は、このプロパティを
	
	`com.pcbsys.foundation.security.auth.fLDAPAdapter`

Nirvana.auth.enabled	`Y` このパラメータが `N` に設定されていて、クレデンシャルが接続要求で渡されると、Universal Messaging サーバはクレデンシャルを無視し、認証なしで接続します。
Nirvana.auth.server.mandatory	`Y`

内部および外部ユーザリポジトリ、Security Infrastructure コマンドラインツールおよび nserver.cnf システムプロパティの詳細については、『*Software AG Infrastructure Administrator's Guide*』および Universal Messaging のマニュアルを参照してください。

- **JAAS フレームワーク** このフレームワークでは、Universal Messaging サーバはユーザクレデンシャルを検証するために、JAAS ログインモジュールを呼び出します。ユーザの認証にカスタムロジックが必要な場合は (外部データベースからユーザクレデンシャルを抽出するためにカスタムサービスが必要なときなど)、SASL よりもこの方式を選択することがあります。Universal Messaging 管理者と協力して以下の項目を用意します。
  - **ログインモジュール**ログインモジュールには、パスワードを抽出して妥当性を検査するコードが含まれます。接続エイリアスで指定されるユーザ名とパスワードの組み合わせがログインモジュールで指定されたものと異なる場合、Integration Server には接続が失敗したことを示すエラーメッセージが表示されます。
  - **JAAS 設定ファイル**Universal Messaging 管理者は、Universal Messaging サーバ上で JASS 設定ファイルを作成する必要があります。このファイルでは JAAS キーを定義し、ログインモジュールを呼び出します。
  - **サーバプロパティ設定**Universal Messaging 管理者は、nserver.conf ファイル (Universal Messaging サーバの *Universal Messaging_directory* \server\umserver\bin\ にあります) でプロパティを以下のように設定する必要があります。

設定するプロパティ	設定値
Nirvana.auth.server.jaaskey	`noauth`
wrapper.java.classpath	ログインモジュールの場所

ログインモジュール、JAAS 設定ファイルおよび nserver.cnf システムプロパティの詳細については、『*Software AG Infrastructure Administrator's Guide*』および Universal Messaging のマニュアルを参照してください。

Broker 接続へのキープアライブモードの指定

Broker への接続を設定した後に、Integration Server が使用するキープアライブモードを指定できます。

キープアライブモードは、クライアントから遮断された接続を Broker でチェックしてからそのクライアントが接続を遮断しているならばクライアントを明示的に切断するかどうかを示します。クライアントを切断すると、Broker ではそのクライアントによって抽出されたすべての未応答ドキュメントを他のクライアントへの再デリバーに使用できるようにします。

注:
キープアライブモードを指定できるのは、Integration Server がバージョン 6.1 以降の webMethods Broker に接続している場合のみです。

クライアント状態が共有でない場合は、接続の切断が検出されなくても、問題にはなりません。Broker は、再接続後、クライアントに未応答イベントを自動的に再デリバーします。ただし、クライアント状態が共有でクライアントが Broker への接続を切断された場合、接続を再確立した後にクライアントは未応答ドキュメントを抽出できません(Integration Server のデフォルトクライアントおよびすべてのトリガークライアントは共有状態クライアントです)。理由は、同一のクライアント ID が共有状態クライアント内ですべてのクライアントによって使用されるからです。Broker は、クライアントの再接続と同一クライアント ID を持つ他のクライアントによる通常の再接続とを区別することができません。今切断されたクライアントによって抽出された未応答ドキュメントは、Broker が明示的な切断通知を受信するとき (通常は TCP/IP 接続がタイムアウトになるとき) まで、再デリバーに使用できません。これには数時間かかることもあります。

未応答ドキュメントが Broker 上で長い時間使用できない状況になることを避けるために、無応答のクライアントを切断して未応答ドキュメントを再デリバーに使用できるようにするキープアライブモードを選択できます。

注:
Broker のキープアライブ機能および共有状態クライアントの詳細については、『*webMethods Broker Java Client API Reference*』を参照してください。

次に示すいずれかのキープアライブモードを設定できます。

- **通常**Broker がキープアライブメッセージを指定された間隔 (キープアライブ間隔) で Integration Server に送信し、別に指定された時間 (最大応答時間) 内は応答を待機します。Broker が応答を受信しない場合は、再試行回数で指定された回数まで再試行します。Integration Server がまだキープアライブメッセージに応答しない場合は、Broker が明示的に Integration Server を切断します。「通常」はデフォルトのモードです。

  たとえば、デフォルトで Broker は Integration Server に 60 秒ごとにキープアライブメッセージを送信します。Integration Server が 60 秒以内に応答しない場合、Broker はキープアライブメッセージを 3 回まで送信した後に Integration Server を切断します(デフォルトの再試行回数は 3 です)。

- **受信専用**Broker は、指定された間隔 (キープアライブ間隔) 内に Integration Server からのアクティビティがない場合は Integration Server を切断します。受信待機モードでは、Broker がキープアライブメッセージを Integration Server に送信せず、再試行回数は無視されます。

たとえば、Broker が Integration Server からのアクティビティを 60 秒待機しているとします。60 秒後に Broker からのアクティビティがない場合、Broker は Integration Server を切断します。

- **無効** Broker が Integration Server へのキープアライブ動作を無効にします。Broker は、キープアライブメッセージを送信せず、非アクティブ状態であることを理由に Integration Server を切断することもありません。

注:
Broker が Integration Server と直接通信することはありません。Broker と Integration Server の間の通信は Broker クライアント API によって実現されます。

キープアライブモードのサーバ設定パラメータ

キープアライブモードは、次に示す一連のサーバ設定パラメータの値の組み合わせで決まります。

- **watt.server.brokerTransport.dur** キープアライブメッセージを Integration Server に送信する前に Broker が待機するアイドル時間の秒数を指定します。これはキープアライブ時間です。デフォルトは 60 秒です。
- **watt.server.brokerTransport.max** Integration Server がキープアライブメッセージに応答するのを Broker が待機する秒数を指定します。これは最大応答時間です。デフォルトは 60 秒です。
- **watt.server.brokerTransport.ret** 応答のない Integration Server を切断する前に Broker がキープアライブメッセージを再送信する回数を指定します。これは再試行回数です。デフォルトは 3 です。

以上のパラメータによるキープアライブモード設定の詳細については、下記を参照してください。以上のパラメータの詳細については、1136 ページの「watt.server.」を参照してください。

通常モード

通常のキープアライブモードを設定するには、次の表に示す設定を使用します。これはデフォルトのモードです。

パラメータ	値
watt.server.brokerTransport.dur	0 より大きくて 2147483647 未満の整数。デフォルトは 60 です。
watt.server.brokerTransport.max	0 より大きくて 2147483647 以下の整数。デフォルトは 60 です。
watt.server.brokerTransport.ret	1～2147483647 の整数。デフォルトは 3 です。

受信待機専用モード

受信専用のキープアライブモードを設定するには、次の表に示す設定を使用します。

パラメータ	値
watt.server.brokerTransport.dur	2147483647
watt.server.brokerTransport.max	0 より大きくて 2147483647 未満の整数。
watt.server.brokerTransport.ret	なし。再試行回数は、受信待機専用モードでは無視されます。

無効

無効のキープアライブモードを設定するには、次の表に示す設定を使用します。

パラメータ	値
watt.server.brokerTransport.dur	2147483647
watt.server.brokerTransport.max	2147483647
watt.server.brokerTransport.ret	1

Integration Server のプライマリポートが変更された場合の Broker クライアントの同期

Integration Server を Broker に接続する前に、サーバ用のプライマリポートを確立しておく必要があります。ただし、Broker 接続エイリアスの設定後に Integration Server のプライマリポート番号を変更した場合、Integration Server の Broker クライアントと Broker の設定との同期が失われる可能性があります。

Broker 接続エイリアスの設定後に Integration Server のプライマリポートを変更した場合は、Broker クライアントを Integration Server の新しいポート設定と同期する必要があります。

Broker クライアントを Integration Server のプライマリポートと同期するには

1. Broker ユーザインタフェースを使用して、サーバの当初のプライマリポート番号、たとえば 10.3.33.129_5555_DefaultClient が反映されているクライアントを削除します。

2. *Integration Server_directory*\instances*instance_name*\config ディレクトリから dispatch.cnf ファイルを削除します。

3. 258 ページの「Broker 接続エイリアスの作成」で説明する手順を使用して、Broker 接続エイリアスを再設定します。

ドキュメントストアの設定

Integration Server は*「ドキュメントストア」*を使用して、ドキュメントの送信中、またはドキュメントが処理を待機している間に、ドキュメントをディスクまたはメモリに保存します。Integration Server は、パブリッシュされたドキュメント用に 3 種類のドキュメントストアを保持しています。

- **デフォルトのドキュメントストア**デフォルトのドキュメントストアには、Integration Server のデフォルト Broker クライアントにデリバーされた保証付きドキュメントが保管されます。Integration Server は、自身のデフォルト Broker クライアントにデリバーされたドキュメントを抽出すると、Integration Server はデフォルトのドキュメントストアにドキュメントを保管します。ディスパッチャがドキュメントをサブスクライブするトリガーを決定するまで、ドキュメントはデフォルトのドキュメントストアに保持されます。次に、ディスパッチャはサブスクライブトリガー用のトリガーキューにドキュメントを移動します。
- **トリガードキュメントストア**トリガードキュメントストアには、特定の webMethods messaging trigger にデリバーされる、ローカルにパブリッシュされた保証付きドキュメントが格納されます。トリガーごとに、Integration Server はトリガードキュメントストア内にローカルにパブリッシュされた保証付きドキュメントのキューを作成します。保証付きドキュメントは、サーバがドキュメントを正常に処理するまで、トリガードキュメントストア内のトリガーキューにあります。
- **送信ドキュメントストア**送信ドキュメントストアは、クライアントサイドキューとも呼ばれ、Broker への送信待ちドキュメントが含まれます。設定済み Integration Server が使用不可能な場合、Broker はドキュメントを送信ドキュメントストアに保管します。Broker への接続が復元されると、Integration Server によって保管されていたドキュメントは Broker に送信され、送信ドキュメントストアは空になります。

  注:
  送信ドキュメントストアは、Broker にパブリッシュされた保証付きドキュメントのみに使用されます。

Integration Server Administrator を使用して、各ドキュメントストアのプロパティを設定できます。

デフォルトのドキュメントストアの設定

デフォルトのドキュメントストアには、Integration Server のデフォルトの Broker クライアントに直接デリバーされたパブリッシュされたドキュメントが保管されます。ディスパッチャがサブスクライブ側の webMethods messaging triggerキューにドキュメントを移動するまで、ドキュメントはデフォルトのドキュメントストアに保持されます。

注:
メッセージングプロバイダとして Universal Messaging を使用する場合は、デフォルトのドキュメントストアを設定する必要はありません。

デフォルトのドキュメントストアを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [リソース] に移動します。
3. [**ストアの設定**] をクリックし、次に [**ドキュメントストアの設定の編集**] をクリックします。
4. [**デフォルトのドキュメントストア**] フィールドを以下のように設定します。

フィールド	説明
[ストアの場所]	デフォルトのドキュメントストアの場所。デフォルトでは、Integration Server はドキュメントストアを次のディレクトリに保存します。 *Integration Server_directory* \instances *instance_name* \DocumentStore デフォルトのドキュメントストアを別の場所に保存する場合、このフィールドでディレクトリを指定します。指定したディレクトリが存在しない場合は、新たに作成されます。 **重要:** 指定したディレクトリに対する書き込みアクセス権を持っていることを確認してください。また、ディレクトリ名にはオペレーティングシステムによって不正と見なされる文字を使用しないでください。
[ストアの初期サイズ]	起動時にデフォルトのドキュメントストアに割り当てるディスク容量。初期サイズを超えるデータを受信すると、デフォルトのドキュメントストアの容量は自動的に増加します。デフォルトのサイズは 25MB です。 [**ストアの初期サイズ**] を設定する際には、デフォルトのドキュメントストアにデリバーされると予想されるドキュメントのサイズと数を考慮する必要があります。ドキュメントのサイズが大きかったり、ドキュメント数が多いと予想される場合、[**ストアの初期サイズ**] の値を大きくすることをお勧めします。 **重要:** Integration Server がインストールされているコンピュータにデフォルトのドキュメントストア、トリガードキュメントストア、および XA 回復ストアの初期サイズを保存するだけの十分な空きディスク容量があることを確認してください。
[容量 (ドキュメント数)]	デフォルトのドキュメントストアに保存できるドキュメントの最大数。デフォルトは 10 ドキュメントです。 [**容量 (ドキュメント数)**] には、[**補充レベル (ドキュメント数)**] よりも大きい値を設定してください。 [**容量 (ドキュメント数)**] を 0 に設定すると、[**補充レベル (ドキュメント数)**] は自動的に一時停止します。[**容量 (ドキュメント数)**] フィール

フィールド	説明
	ドを 0 に設定すると、[**設定**] > [リソース] > [ストアの設定] ページで、フィールドの横に [一時停止中] と表示されます。 **注:** サーバ用に設定されている Broker が存在しない場合、[**容量 (ドキュメント数)**] フィールドに [Broker 未設定] と表示されます。
[補充レベル (ドキュメント数)]	Integration Server が Broker から次のドキュメント抽出を開始するまでにデフォルトのドキュメントストアに残っている未処理ドキュメントの数。 たとえば、デフォルトのドキュメントストアの [**容量 (ドキュメント数)**] を 10 に設定し、[**補充レベル (ドキュメント数)**] を 4 に設定すると、最初に 10 個のドキュメントが抽出されます。デフォルトのドキュメントストアに残っている未処理ドキュメントの数が 4 つになると、さらに 6 つのドキュメントが抽出されます。6 つのドキュメントが使用できない場合には、使用可能な数だけのドキュメントが抽出されます。 デフォルトでは、補充レベルは 4 つのドキュメントに設定されています。 [**補充レベル (ドキュメント数)**] には、[**容量 (ドキュメント数)**] よりも小さい値を設定してください。[**容量 (ドキュメント数)**] を 0 に設定すると、[**補充レベル (ドキュメント数)**] は自動的に一時停止します。 **注:** サーバ用に設定されている Broker が存在しない場合、[**補充レベル (ドキュメント数)**] フィールドに [Broker 未設定] と表示されます。

5. [**変更内容の保存**] をクリックします。

6. 新規の値を有効にするにはサーバの再起動が必要なパラメータを変更した場合は、Integration Server を再起動します。

注:
パラメータの横にアスタリスク (*) が付いている場合、変更内容を有効にするには再起動が必要です。

トリガードキュメントストアについて

トリガーキュメントストアには、Integration Server が処理の待機のためにローカルにパブリッシュされた保証付きドキュメントを保持するトリガーキューが格納されます。ドキュメントは、以下のいずれかの状態が発生するまで、トリガードキュメントストア内のトリガーキューにあります。

- トリガー条件がドキュメントによって満たされ、その条件で指定されているトリガーサービスが Integration Server によって正常に実行される。
- ドキュメントがなんらかのトリガー条件を満たさないために、そのドキュメントが Integration Server によって廃棄される。
- ドキュメントがトリガーによる処理済みの重複ドキュメントであるため、Integration Server によって廃棄される。これは、トリガーに対して重複抑制処理が有効に設定されている場合にのみ発生します。
- トリガーによる処理済みのドキュメントであるかどうかを判断できないため、インダウトドキュメントと見なされ、そのドキュメントのログをとるよう Integration Server から監査サブシステムに通知される。これは、トリガーに対して重複抑制処理が有効に設定されている場合にのみ発生します。

トリガードキュメントストアの設定

トリガードキュメントストアを設定する際には、ドキュメントストアの場所や初期サイズを設定できます。

注:
メッセージングプロバイダとして Universal Messaging を使用する場合は、トリガードキュメントストアを設定する必要はありません。

トリガードキュメントストアを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [リソース] に移動します。
3. [**ストアの設定**] をクリックし、次に [**ドキュメントストアの設定の編集**] をクリックします。
4. [**トリガードキュメントストア**] フィールドを以下のように設定します。

フィールド	説明
[ストアの場所]	トリガードキュメントストアの場所。デフォルトでは、Integration Server はトリガードキュメントストアを次のディレクトリに保存します。 *Integration Server_directory* \instances*instance_name*\DocumentStore トリガードキュメントストアを別のディレクトリに保存する場合、このフィールドでディレクトリを指定します。指定したディレクトリが存在しない場合は、新たに作成されます。 **重要:**

フィールド	説明
	指定したディレクトリに対する書き込みアクセス権を持っていることを確認してください。また、ディレクトリ名にはオペレーティングシステムによって不正と見なされる文字を使用しないでください。
[ストアの初期サイズ]	起動時にトリガードキュメントストアに割り当てるディスク容量。初期サイズを超えるデータを受信すると、トリガードキュメントストアの容量は自動的に増加します。デフォルトのサイズは 35MB です。 **重要:** Integration Server がインストールされているコンピュータにデフォルトのドキュメントストア、トリガードキュメントストア、および XA 回復ストアの初期サイズを保存するだけの十分な空きディスク容量があることを確認してください。

5. [**変更内容の保存**] をクリックします。

6. 新規の値を有効にするにはサーバの再起動が必要なパラメータを変更した場合は、Integration Server を再起動します。

注:
パラメータの横にアスタリスク (*) が付いている場合、変更内容を有効にするには再起動が必要です。

受信ドキュメントの履歴の保持

Integration Server をバージョン 6.0.1 の Broker に接続する場合、サーバで受信したドキュメントの履歴を保持するよう [**受信ドキュメントの履歴**] を設定できます。この設定によって、ごく基本的な形の重複検出がすべてのトリガーに対して実行されます。

Integration Server をバージョン 6.1 以降の Broker に接続する場合、重複検出をトリガー単位で設定できます。バージョン 6.1 以降の Integration Server を使用して webMethods messaging triggerに対して重複検出を設定する方法については、『*webMethods Service Development Help*』を参照してください。

Integration Server のクラスタをバージョン 6.0.1 の Broker に接続する場合、クラスタ内の Integration Server ごとに独自の受信ドキュメントの履歴情報を保持します。つまり、受信ドキュメントの履歴情報は、クラスタ内の Integration Server 間で共有されません。

注:
[**受信ドキュメントの履歴 (分)**] フィールドは、Integration Server をバージョン 6.0.1 の Broker に接続する場合にのみ設定できます。バージョン 6.1 以降の Broker に接続する場合、このフィールドは設定できません。受信ドキュメントの履歴の設定の詳細については、バージョン 6.0.1 の『*webMethods Integration Server Administrator's Guide*』を参照してください。

受信クライアントサイドキューの有効化

Integration Server をバージョン 6.0.1 の Broker に接続する場合、受信クライアントサイドキューを使用できます。受信クライアントサイドキューを有効に設定すると、Integration Server は受信ドキュメントをディスクに保存し、ドキュメントの受信と保存が完了した直後に Broker に対して通知を行います。受信クライアントサイドキューを無効に設定すると、Integration Server は受信ドキュメントをメモリに保存し、ドキュメントの処理が完了した後に webMethods Broker に対して通知を行います。

注:
Integration Server をバージョン 6.1 以降の Broker に接続する場合、受信クライアントサイドキューは使用できません。バージョン 6.0.1 の Broker に接続してクライアントサイドキューを使用する方法については、バージョン 6.0.1 の『*webMethods Integration Server Administrator's Guide*』を参照してください。

送信ドキュメントストアについて

送信ドキュメントストアは、クライアントサイドキュー (CSQ) とも呼ばれ、Broker 接続エイリアスで指定された Broker が使用できないときに Integration Server によってパブリッシュされた保証付きドキュメントを含みます。Broker への接続が再確立されると、Integration Server によって送信ドキュメントストア内のドキュメントは Broker に送信されます。

注:
送信ドキュメントストアは、Universal Messaging にパブリッシュされた保証付きドキュメントに使用されません。代わりに、それぞれの Universal Messaging 接続エイリアスには専用のクライアントサイドキューがあります。Universal Messaging のクライアントサイドキューの設定は、Universal Messaging 接続エイリアスの作成で行います。Universal Messaging 接続エイリアスの作成の詳細については、262 ページの「Universal Messaging 接続エイリアスの作成」を参照してください。

Integration Server が送信ドキュメントストアを使用するには、watt.server.publish.useCSQ パラメータを「always」に設定する必要があります。watt.server.publish.useCSQ パラメータが「never」に設定された場合、Integration Server は、ドキュメントが Broker にパブリッシュされるときに Broker が使用不可能な場合に、ServiceException をスローします。

CSQ から Broker へドキュメントをパブリッシュする最初の試行が失敗した場合、Integration Server はドキュメントが正常にパブリッシュされるまで、または、watt.server.publish.maxCSQRedeliveryCount で指定された最大試行回数に達するまで再試行を実行します。CSQ から Broker にパブリッシュする各試行は再デリバー試行と見なされます。Integration Server が CSQ から Integration Server へのドキュメント送信を指定された試行回数実行し、すべての試行が失敗した後、監査サブシステムはドキュメントのログして、状態に STATUS_TOO_MANY_TRIES を割り当てます。

Integration Server では、パブリケーションの順序で、または並行して送信ドキュメントストアのドキュメントを空にできます。送信ドキュメントストアが空になる方法は、watt.server.publish.drainCSQInOrder パラメータの値によって決定されます。デフォルトでは、Integration Server は送信ドキュメントストアが空になるまで、新規にパブリッシュされたすべてのドキュメント (保証付きドキュメントおよび揮発性ドキュメント) を送信ドキュメントストアに送信します。この処理を行うことで、Integration Server はパブリケーション順序を維持で

きます。送信ストアを並行して空にするように Integration Server を設定すると、新規ドキュメントが Broker にパブリッシュされる間に送信ストアが空になります。

Integration Server が送信ドキュメントストアを排出する速度の設定

サーバが送信ドキュメントストアを空にする速さを設定するには、[**設定**] > [**リソース**] > [**ストアの設定**] > [**ドキュメントストアの設定の編集**] ページで、[**各トランザクションで送信するドキュメントの最大数**] パラメータを設定します。デフォルトでは、このパラメータは 25 個のドキュメントに設定されています。送信ドキュメントストアをより速く空にするには、各トランザクションで送信するドキュメントの数を増やします。各トランザクションで送信するドキュメントの数を増やすと、その分だけ必要なメモリ量が多くなることに注意してください。送信ドキュメントストアを空にする際にメモリが多く消費されないようにする場合や、送信ドキュメントストアを空にする速さが遅くてもかまわない場合には、各トランザクションで送信するドキュメントの数を減らします。ただし、Integration Server から Broker にドキュメントを直接パブリッシュする場合に、迅速に実行させたり使用されるリソースを少なくしたりするために、送信ドキュメントストアの排出をできるだけ速くすることをお勧めします。

以下の場合に限り、Integration Server が送信ドキュメントストア (クライアントサイドキュー) を排出する速度を設定する必要があります。

- Integration Server がドキュメントを Broker にパブリッシュする場合。Integration Server は、ドキュメントを Universal Messaging にパブリッシュするときに送信ドキュメントストアを使用しません。
- 送信ドキュメントストアが Integration Server に対して有効である。つまり、watt.server.publish.useCSQ パラメータが「always」に設定されている場合。

ヒント:
送信ドキュメントストア内のドキュメントの数を監視するには、[**メッセージング**] > [webMethods の設定] ページで、Broker 接続エイリアスの [**CSQ カウント**] フィールドの値をチェックします。

Integration Server が送信ドキュメントストアを空にする速さを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [**リソース**] に移動します。
3. [**ストアの設定**] をクリックし、次に [**ドキュメントストアの設定の編集**] をクリックします。
4. [**送信ドキュメントストア**] の [**各トランザクションで送信するドキュメントの最大数**] に、各トランザクションで送信ドキュメントストアから Broker に送信するドキュメントの数を入力します。

   設定済みの Broker が存在しない場合には、Integration Server Administrator のフィールド名の横に [Broker 未設定] と表示されます。
5. [**変更内容の保存**] をクリックします。

送信ドキュメントストアの容量の設定

デフォルトでは、送信ドキュメントストア (クライアントサイドキュー) には最大 500,000 個のドキュメントを保管できるように設定されます。送信ドキュメントストアの最大容量に達すると、ドキュメントのパブリッシュサービスを実行するスレッドは、ブロックされるか、または一時停止します。送信ドキュメントストアの排出が開始されるまで、スレッドはブロックされたままになります。

送信ドキュメントストアの容量は、watt.server.control.maxPersist サーバパラメータによって決まります。Broker を長期間停止する場合、このパラメータを編集して、送信ドキュメントストアの容量を小さく設定することをお勧めします。送信ドキュメントストアの容量をデフォルト設定のまま使用すると、Broker が使用不可能になったときに、送信ドキュメントを保管するためにメモリが多く消費され、Integration Server でエラーが発生する可能性があります。送信ドキュメントストアの容量を小さく設定すると、Integration Server はスレッドをブロックするので、ドキュメントを保管するために多くのメモリを消費し続けることがなくなります。

ユーザアカウントと webMethods Messaging Triggerサービスの関連付け

HTTP 要求を介してクライアントからサービスを呼び出すと、Integration Server はそのサービスに割り当てられた実行 ACL に対してクライアントのユーザグループのメンバーシップおよびクレデンシャルを確認します。Integration Server はこのチェックを実行して、クライアントがそのサービスを呼び出す許可を持っていることを確認します。ただし、パブリッシュ/サブスクライブ状態では、クライアント要求によらず、ドキュメントを Integration Server が受信した時点で、サービスが呼び出されます。Integration Server ではユーザのクレデンシャルはパブリッシュ済みドキュメントと関連付けられないため、webMethods messaging triggerに関連付けられたサービスを呼び出すときに Integration Server が使用するユーザアカウントを指定することができます。

事前定義済みのユーザアカウント (Administrator、Central、Default、Developer、Replicator) のいずれか 1 つのクレデンシャルを使用して、Integration Server がサービスを呼び出すように設定できます。また、管理者または別の定義済みサーバ管理者のユーザアカウントを指定することもできます。トリガー条件を満たすドキュメントを Integration Server が受信した場合、Integration Server は指定されたユーザアカウントのクレデンシャルを使用して、トリガー条件で指定されたサービスを呼び出します。

選択したユーザアカウントには、トリガーに関連付けられているサービスに割り当てられた実行 ACL に必要なクレデンシャルが含まれていることを確認してください。たとえば、トリガーでサービスを呼び出すユーザアカウントを Developer に設定したとします。receiveCustomerInfo トリガーには、パブリッシュ可能なドキュメントタイプを addCustomer サービスに関連付けるという条件が含まれています。また、addCustomer サービスで、実行 ACL が Replicator に設定されているとします。トリガー条件が満たされたときに、選択したユーザ設定 (Developer) はサービスの呼び出しに必要なクレデンシャル (Replicator) を持たないため、addCustomer サービスは実行されません。

webMethods messaging trigger サービスのユーザアカウントを指定する方法は、トリガによって使用されるメッセージングプロバイダによって異なります。

- トリガーが Broker からのメッセージを受信する場合は、Integration Server Administrator を使用してユーザアカウントを指定します。指定するユーザアカウントは、Brokerからメッセージを受信するすべての webMethods messaging trigger に適用されます。詳細については、294 ページの「webMethods Messaging Triggerサービスを呼び出すためのユーザアカウントの指定」を参照してください。
- トリガーが Universal Messaging からのメッセージを受信する場合は、Designer を使用してトリガーのユーザアカウントを指定します。Universal Messaging からメッセージを抽出する webMethods messaging triggerごとに [**実行ユーザ**] プロパティを使用して、そのトリガーのトリガーサービスを呼び出すことができるユーザを指定します。トリガーの実行ユーザの設定の詳細については、*webMethods Service Development Help*を参照してください。

webMethods Messaging Triggerサービスを呼び出すためのユーザアカウントの指定

Broker からメッセージを受信する webMethods messaging triggerについて、Broker を使用して実行ユーザを設定します。指定するユーザアカウントは、Broker からメッセージを受信するすべての webMethods messaging trigger に適用されます。

webMethods messaging triggerサービスを実行するユーザアカウントを指定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [リソース] に移動します。
3. [**ストアの設定**] をクリックし、次に [**ドキュメントストアの設定の編集**] をクリックします。
4. [**トリガードキュメントストア**] 領域の [**ユーザ**] フィールドで、Integration Server がトリガー条件で指定されたサービスの実行に使用するクレデンシャルユーザアカウントを選択します。中央ディレクトリまたは外部ディレクトリからユーザアカウントを選択できます。デフォルトは Administrator です。
5. [**変更内容の保存**] をクリックします。

Integration Server の非クラスタグループとの負荷分散

Integration Server では、Integration Server が非クラスタグループとしてメッセージングプロバイダと接続している場合に、負荷分散された方法で同一メッセージングプロバイダからのメッセージを受信できます。非クラスタグループでは、複数の Integration Server がクラスタのメンバーではない単一のメッセージクライアントとして機能します。

注:
「非クラスタグループ」という用語のほかに、「ステートレスクラスタ」および「外部クラスタ」という用語が使用されることがあり、それらはIntegration Server のグループがクラスタと同様に機能する一方で、設定されたクラスタの一部ではないという状況を表します。

Integration Server では、Integration Server が非クラスタグループとしてメッセージングプロバイダと接続している場合に、負荷分散された方法で同一メッセージングプロバイダからのメッセージを受信できます。

負荷分散された方法でメッセージを受信するには、Integration Server ごとに次の条件を満たす必要があります。

- **同一のメッセージングプロバイダに接続する**非クラスタ Integration Server グループは、負荷分散された方法で Universal Messaging または Broker からメッセージを受信します。メッセージング接続エイリアスは、Integration Server ごとに同一である必要があります。
- **同一のクライアントプリフィックスを使用する**Integration Server Administrator を使用して、Integration Server のクライアントプリフィックスを指定します。詳細については、258 ページの「Broker 接続エイリアスの作成」および262 ページの「Universal Messaging 接続エイリアスの作成」を参照してください。
- **同一のクライアントグループを使用する**Broker がメッセージングプロバイダである場合は、各 Integration Server が同一のクライアントグループを使用する必要があります。Integration Server Administrator を使用して、Integration Server のクライアントグループを指定します。詳細については、258 ページの「Broker 接続エイリアスの作成」を参照してください。
- **同一のトリガーを使用する**負荷分散された方法でメッセージを受信するには、非クラスタ Integration Server は、単一のメッセージングクライアントとして機能する必要があります。これには、同一のドキュメントサブスクリプションを持つことも含まれます。webMethods messaging trigger の設定については、*webMethods Service Development Help*を参照してください。

非クラスタグループ内の Integration Server も、Integration Server クラスタ内になくてもドキュメント履歴データベースおよび相互参照データベースを共有できます。

注:
Integration Server のステートレスクラスタでは Terracotta Server Array を使用しません。

webMethods メッセージングでのステートレスクラスタ使用上の重要な考慮事項

負荷分散のために Integration Server の非クラスタグループと同一のメッセージングプロバイダを接続するかどうかを決定するときは、以下の点に留意してください。

- グループ内の Integration Server は、同一バージョンおよび同一修正レベルである必要があります。
- グループ内の Integration Server は、ドキュメント履歴データベースおよび相互参照データベースを共有できます。
- Integration Server が Integration Server の非クラスタグループのトリガーを変更するときに実行するアクションは[**クライアントプリフィックスの共有**]オプションの値によって異なります。
  - メッセージングプロバイダへの Integration Server 接続の [**クライアントプリフィックスの共有**] オプションが [**はい**] に設定されている場合、Integration Server はメッセージングプロバイダの共有オブジェクトを更新しません。

[**クライアントプリフィックスの共有**] オプションが [**はい**] である Integration Server の非クラスタグループでは、メッセージングプロバイダの設定を自ら手動で更新する必要があります。

- メッセージングプロバイダへの Integration Server 接続の [**クライアントプリフィックスの共有**] オプションが [**いいえ**] に設定されている場合、Integration Server はキュー、チャネルなど、メッセージングプロバイダの共有オブジェクトの更新を許可します。Integration Server の 1 つで webMethods messaging triggerを変更すると、メッセージングプロバイダのドキュメントおよび他のデータが削除されることがあります。たとえば、メッセージングプロバイダが Broker であるときにトリガーを有効にしたり、トリガーを無効にしたり、処理モードを変更したりすると、Integration Server は Broker で関連するトリガークライアントキューを削除してから再作成します。このアクションによって、Broker 上のトリガークライアントキュー内にあるドキュメントもすべて削除されます。

14 JMS メッセージングのための Integration Server の設定

- JMS メッセージング設定の概要 298
- JNDI プロバイダの使用 298
- JMS 接続エイリアスの使用 305
- 管理オブジェクトの作成 329
- 接続ファクトリオブジェクトの変更の監視 331
- JMS での SSL の使用 336
- サポートされている JMS プロバイダ 336
- Integration Server クラスパスへの JMS プロバイダクライアントライブラリの追加 341

JMS メッセージング設定の概要

Integration Server を JMS メッセージ用に設定するには、次の作業を実行する必要があります。

- 1 つ以上の JNDI プロバイダエイリアスを作成して、Integration Server で JMS プロバイダへの接続を作成するかメッセージを送受信するための宛先を指定する必要がある場合に、Integration Server で管理対象オブジェクトを検索できる場所を指定します。
- 1 つ以上の接続エイリアスを作成して、JMS プロバイダとの接続を確立するために Integration Server に必要なプロパティをカプセル化します。

また、Integration Server にはさまざまなサーバ設定プロパティが含まれており、これらのプロパティを使用してデフォルトのサーバ動作を変更することもできます。JMS に関連するサーバ設定パラメータの一覧については、1095 ページの「サーバ設定パラメータ」を参照してください。

JNDI プロバイダの使用

各 JMS プロバイダは、JNDI (Java Naming and Directory Interface) と呼ばれる標準ネームスペースに JMS 管理対象オブジェクトを保存できます。JNDI は、Java アプリケーションにネーミングおよびディレクトリの機能を提供する Java API です。

JMS クライアントと同様に、Integration Server は JNDI プロバイダエイリアスを使用して、管理対象オブジェクトの検索に必要な情報をカプセル化します。JMS 接続エイリアスの作成時に、Integration Server によって管理対象オブジェクト (接続ファクトリ、宛先など) の検索に使用される JNDI プロバイダエイリアスを指定できます。

注:
ネイティブ webMethods API を使用して直接 webMethods Broker に接続する場合は、JNDI プロバイダを使用する必要はありません。webMethods Broker は使用するけれども (ネイティブ webMethods API を使用した) ネイティブな接続は行わない場合は、今までどおり JNDI プロバイダを使用する必要があります。ネイティブ webMethods API を使用して webMethods Broker に接続する方法の詳細については、306 ページの「ネイティブ webMethods API を使用した webMethods Broker への接続」を参照してください。

事前定義済みの JNDI プロバイダエイリアス

Integration Server には、事前定義済みの JNDI プロバイダエイリアスがあります。Integration Server を最初に起動したときに、Integration Server によってエイリアスが作成されます。以前のバージョンから移行した Integration Server にエイリアスが存在していた場合、Integration Server には既存のエイリアスが保持されます。つまり、Integration Server では以前のエイリアス定義は上書きされません。

Integration Server の事前定義済みの JNDI プロバイダエイリアスは DEFAULT_IS_JNDI_PROVIDER です。Software AG Universal Messaging のローカルインスタンスへの接続を確立するための設定が事前定義された JNDI プロバイダエイリアスです。Software AG Universal Messaging が Integration Server と同じディレクトリにインストールされていない場合、DEFAULT_IS_JNDI_PROVIDER は nsp://localhost:9000 のプロバイダ URL を使用します。

注:

以前のバージョンから Integration Server バージョン 9.10 以降に移行した場合は、EventBusJndiProvider エイリアスがある場合があります。これは 9.10 より前のバージョン用の事前定義済みの JNDI プロバイダエイリアスでした。このエイリアスは、Software AG Universal Messaging のローカルインスタンスを指し示します。Software AG Universal Messaging が Integration Server と同じディレクトリにインストールされていない場合、EventBusJndiProvider は nsp://localhost:9000 のプロバイダ URL を使用します。

JNDI プロバイダエイリアスの作成

JNDI プロバイダへのエイリアスを作成するには、以下の手順に従います。

JNDI プロバイダエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JNDI 設定] に移動します。
3. [**JNDI プロバイダエイリアスの作成**] をクリックします。
4. JNDI プロバイダエイリアスに関する次の情報を指定します。

フィールド	説明
[JNDI エイリアス名]	この JNDI プロバイダに割り当てるエイリアス名。
説明	この JNDI エイリアスの説明。
[定義済み JNDI テンプレート]	使用する JNDI テンプレート。 JNDI テンプレートにより、特定のプロバイダのエイリアス設定を完了するための情報が得られます。 **注:** JNDI プロバイダを作成すると、Integration Server Administrator によって、現在の設定が [**定義済み JNDI テンプレート**] フィールドの値として表示されます。すなわち、Integration Server によって、現在指定されている設定が JNDI プロバイダエイリアスに使用されます。
[初期コンテキストファクトリ]	JNDI プロバイダのクラス名。JNDI プロバイダは、ネーミングおよびディレクトリ操作の名前解決の開始点として初期コンテキストを使用します。 定義済み JNDI テンプレートを選択した場合は、Integration Server によってプロバイダの初期コンテキストファクトリが表示されます。

フィールド	説明
[プロバイダ URL]	JNDI プロバイダを使用したセッションの初期コンテキストのプライマリ URL。URL は、JNDI プロバイダによって JMS 管理対象オブジェクトが保存される JNDI ディレクトリを指定します。 定義済み JNDI テンプレートを選択した場合は、カッコ << >> 内のプレースホルダ情報を各自の設定に固有の情報で置き換えます。 Software AG Universal Messaging を使用する場合、これは Universal Messaging 領域サーバであり、形式は nsp://*UM_host*:*UM_port* (nsp://127.0.0.1:9000 など) となります。 Universal Messaging 領域サーバのクラスタを使用する場合は、クラスタ内の各領域サーバの URL のリストを入力します。各 URL を区切るには、コロンまたはセミコロンを使用します。 ■ Integration Server が常にリストの最初の Universal Messaging サーバに接続を試みる場合は、カンマを使用して URL を区切ります。リストの 2 番目の Universal Messaging サーバに接続を試みるのは、最初のサーバが使用不可になった場合だけです。 ■ Integration Server がリストからランダムに選択された URL に接続する場合は、セミコロンを使用して URL を区切ります。この場合は、クライアントがクラスタ内のサーバに適度に分散します。 **注:** クラスタ内の各 Universal Messagingサーバの URL を指定する必要があります。Integration Server は、URL が指定されている Universal Messaging サーバのみに接続します。その後、Integration Server は、指定された Universal Messaging サーバで宛先、継続的サブスクライバなどのリソースを作成します。クラスタで Universal Messaging サーバを 1 台だけ指定すると、リソースはサーバのみに対してローカルになります。 **注:** JMS 接続エイリアスがこの JNDI プロバイダエイリアスを使用して Universal Messaging クラスタに接続し、JMS 接続エイリアスがマスターを追跡するように設定されている場合、Integration Server は常にクラスタのマスター領域サーバに接続します。デフォルトで、Integration Server はクラスタのマスター領域サーバを追跡するよう設定されます。Software AG は、JMS 接続エイリアスがマスターを追跡する場合、URL のカンマ区切りリストを使用することをお勧めします。JMS 接続エイリアスの [**マスター追跡の有効化**] オプションの詳細に

フィールド	説明
	ついては、以下を参照してください。307 ページの「JMS 接続エイリアスの作成」
[プロバイダ URL フェイルオーバーリスト]	JNDI プロバイダを使用したセッションの初期コンテキストのフェイルオーバー URL のリスト。行ごとに URL を 1 つ指定します。プライマリ JNDI プロバイダへの接続が使用不可になった場合、Integration Server は自動的にこのリストに指定されている JNDI プロバイダへの接続を試行します。 JNDI プロバイダフェイルオーバーリストの設定の詳細については、303 ページの「JNDI プロバイダフェイルオーバーリストの作成」を参照してください。
[セキュリティプリンシパル]	JNDI プロバイダが JNDI ディレクトリへのアクセスにプリンシパル名を必要とする場合に、Integration Server によって JNDI プロバイダに提供されるプリンシパル名 (ユーザ名)。 JNDI プロバイダにセキュリティプリンシパル情報が必要かどうかは、JNDI プロバイダの製品資料を参照してください。
[セキュリティクレデンシャル]	JNDI プロバイダが JNDI ディレクトリへのアクセスにセキュリティクレデンシャルを必要とする場合に、Integration Server によって JNDI プロバイダに提供されるクレデンシャル (パスワード)。 JNDI プロバイダにセキュリティクレデンシャルが必要かどうかは、JNDI プロバイダの製品資料を参照してください。
[その他のプロパティ]	JNDI プロバイダの設定に必要なその他のプロパティ。たとえば、JNDI プロバイダが JNDI への接続に必要とする追加の .jar ファイルまたはクラスファイルのクラスパスを指定することが必要になる場合もあります。 定義済み JNDI テンプレートを選択した場合、Integration Server によってこのフィールドには JNDI プロバイダに必要なその他のプロパティおよびプレースホルダ情報が移入されます。 JNDI プロバイダに必要なその他のプロパティまたはクラスやこれらのファイルの場所の詳細については、JNDI プロバイダの製品資料を参照してください。

5. [**SSL の使用**]: Integration Server と JNDI プロバイダの間でセキュアな通信を確立する場合、[**はい**] を選択し、[**SSL 設定**] および [**JNDI プロパティ名 (プロバイダ固有)**] のパラメータを設定します。デフォルトで、値は [**いいえ**] です。

6. [**SSL 設定**] で、以下を指定します。

フィールド	説明
[トラストストアエイリアス]	JNDI プロバイダの認証局 (CA) の認証を含むトラストストアのエイリアス。 SSL通信を設定する場合、トラストストアエイリアスまたはキーストアエイリアスを選択する必要があります。
[キーストアエイリアス]	JNDI プロバイダへの接続時に Integration Server で使用するクライアント認証を含むキーストアのエイリアス。
[キーエイリアス]	JNDIプロバイダに接続するプライベートキーを含むキーのエイリアス。このキーエイリアスは、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

7. [**JNDI プロパティ名 (プロバイダ固有)**] で、以下を指定します。

フィールド	説明
[トラストストアのプロパティ名]	トラストストアの場所を保存する JNDI プロパティ名。 [**SSL 設定**] の [**トラストストアエイリアス**] を設定する場合、トラストストアの場所の JNDI プロパティ名は必須です。
[トラストストアのパスワードのプロパティ名]	トラストストアのパスワードを保存する JNDI プロパティ名。 [**SSL 設定**] の [**トラストストアエイリアス**] を設定する場合、トラストストアのパスワードの JNDI プロパティ名は必須です。
[キーストアのプロパティ名]	キーストアの場所を保存する JNDI プロパティ名。 [**SSL 設定**] の [**キーストアエイリアス**] を設定する場合、キーストアの場所の JNDI プロパティ名は必須です。
[キーストアのパスワードのプロパティ名]	キーストアのパスワードを保存する JNDI プロパティ名。 [**SSL 設定**] の [**キーストアエイリアス**] を設定する場合、キーストアのパスワードの JNDI プロパティ名は必須です。
[キーストアフォーマットのプロパティ名]	キーストアの形式を保存する JNDI プロパティ名。このフィールドはオプションです。
[プライベートキープロパティ名]	プライベートキー名を保存する JNDI プロパティ名。このフィールドはオプションです。

8. [**変更内容の保存**] をクリックします。

JNDI プロバイダエイリアスの編集

既存の JNDI プロバイダエイリアスを編集するには、以下の手順に従います。

JNDI プロバイダエイリアスを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JNDI 設定] に移動します。
3. [JNDI プロバイダエイリアスの定義] リストから、編集する JNDI プロバイダエイリアスを選択します。Integration Server Administrator によって、JNDI プロバイダエイリアスの詳細が表示されます。
4. [**JNDI プロバイダエイリアスの編集**] をクリックします。
5. JNDI プロバイダエイリアスのプロパティを編集します。

   フィールドの詳細については、299 ページの「JNDI プロバイダエイリアスの作成」を参照してください。
6. [**変更内容の保存**] をクリックします。

JNDI プロバイダエイリアスの削除

JNDI プロバイダエイリアスを削除するには、以下の手順に従います。

重要:
JNDI プロバイダエイリアスを削除すると、その JNDI プロバイダエイリアスに依存している JMS 接続エイリアスが使用されているサービスまたは JMS トリガーでエラーが発生します。

JNDI プロバイダエイリアスを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JNDI 設定] に移動します。
3. 削除するエイリアスを検索し、[削除] 列の ✕ アイコンをクリックします。Integration Server によって、アクションの確認を求めるダイアログボックスが表示されます。[**OK**] をクリックして、JNDI プロバイダエイリアスの削除を確認します。

JNDI プロバイダフェイルオーバーリストの作成

プライマリ JNDI プロバイダへの接続が使用不可になった場合に自動的に代替の JNDI プロバイダに接続するように Integration Server を設定できます。

[**JNDI プロバイダエイリアスの編集**] ページを使用して、プライマリ接続が失敗した場合に Integration Server が使用できる代替 URL を設定した JNDI プロバイダフェイルオーバーリストを作成します。

JNDI プロバイダ URL をフェイルオーバーリストに追加するときは、以下の点に留意してください。

- JNDI プロバイダはプライマリプロバイダと同じタイプである必要があります。たとえば、プライマリプロバイダが webMethods Broker である場合は、フェイルオーバーリスト内の JNDI プロバイダも webMethods Broker である必要があります。
- プロバイダの管理対象オブジェクトが互いに同一である必要があります。
- Integration Server は JNDI プロバイダに接続した後、接続が失われるか中断されるまでその JNDI プロバイダを使用し続けます。
- 接続エイリアスを開始または再開すると、Integration Server はプライマリ JNDI プロバイダへの接続を試行します。この接続に失敗すると、Integration Server は直ちにフェイルオーバーリスト内の最初の JNDI プロバイダへの接続を試行します。さらにこの接続にも失敗すると、Integration Server はリスト内の次の JNDI プロバイダへの接続を試行します。
- JNDI プロバイダとして webMethods Broker を使用する場合は、webMethods Broker テリトリを使用すると webMethods Broker 間でオブジェクトの同期を保つことができます。このようにすると、テリトリ内の別の webMethods Broker へオブジェクトが自動的に転送されます。
- Universal Messaging 領域サーバのクラスタを JNDI プロバイダとして使用する場合は、Software AG は、[**プロバイダ URL フェイルオーバーリスト**] 値を使用しないことをお勧めします。[**プロバイダ URL**] 機能で指定される領域 URL は、フェイルオーバーリストとして機能します。

JNDI プロバイダに対するテスト検索の実行

Integration Server Administrator を使用すると、プライマリ JNDI プロバイダおよび [**プロバイダ URL フェイルオーバーリスト**] で指定したフェイルオーバー JNDI プロバイダに対してテスト検索を実行できます。テスト検索は、URL が有効であるかどうかを検証します。

注:
テスト検索は URL の妥当性のみを検証し、指定したプロバイダ URL エイリアスによって参照される JNDI ネームスペース内のオブジェクトのチェックおよび比較は行いません。

JNDI プロバイダに対してテスト検索を実行するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JNDI 設定] に移動します。
3. テストするエイリアスを検索し、[テスト検索] 列の ▶ をクリックします。

   Integration Server によって、JNDI ネームスペース内にあるオブジェクトのリストが返されます。

   注:

リストには、最初に成功したプロバイダエイリアス URL 検索で参照された JNDI ネームスペース内のオブジェクトが一覧表示されます。つまり、表示されている結果は、プライマリエイリアスの結果である場合もあれば、そうではなくフェイルオーバーエイリアスの結果である可能性もあります。

管理対象オブジェクトに対する JNDI プロバイダのキャッシュおよびタイムアウト動作

JMS 接続エイリアスが有効な場合、Integration Server は JNDI を通じて管理対象オブジェクトを抽出します。管理対象オブジェクトを抽出するために、Integration Server は指定された JNDI プロバイダを使用して新しいコンテキストオブジェクトをインスタンス化します。Integration Server はこのコンテキストを使用して、JMS 接続エイリアスを使用する JMS トリガー、メッセージ送信サービスおよびメッセージ受信サービスで指定されている接続ファクトリおよび宛先を検索します。

コンテキストがインスタンス化されると、webMethods Broker 実装を含む大部分の JNDI プロバイダ実装で、自動的に管理対象オブジェクトがローカルにキャッシュされます。ただし、一部の JNDI プロバイダでは、管理対象オブジェクトのキャッシュおよびタイムアウト動作をプロバイダで設定することができます。ほとんどの場合、宛先は検索されたときに Integration Server によってキャッシュされるため、このタイプのプロバイダを使用しても問題はありません。Integration Server は宛先を 1 度のみ検索する必要があり、後続の検索はすべてローカルに抽出されます。ただし、管理対象オブジェクトがコンテキストにキャッシュされていない場合またはタイムアウトになっている場合は、検索要求によって JNDI プロバイダへの新しい接続が要求されることがあります。

この場合、接続エイリアスを有効にしてから宛先を抽出するまでの間に JNDI プロバイダが使用不可になると、NamingException が発生する可能性があります。この状況は、Integration Server の JNDI フェイルオーバー機能で対処されます。Integration Server で元のコンテキストから宛先を抽出できない場合、いずれかのフェイルオーバー JNDI プロバイダから新しいコンテキストを作成します。

Integration Server は宛先をローカルにキャッシュするため、キャッシュされた宛先が失効状態になる可能性があることに注意してください。この場合には、元の宛先が使用されます。接続エイリアスを有効にして無効にするか、接続オブジェクトの変更を監視することで、管理対象オブジェクトを再ロードできます。管理対象オブジェクトが使用不可になると、ServiceException が発生します。接続の監視の詳細については、331 ページの「接続ファクトリオブジェクトの変更の監視」を参照してください。

JMS 接続エイリアスの使用

JMS 接続エイリアスは、Integration Server と JMS プロバイダ間にアクティブな接続を確立するために Integration Server に必要な情報を指定します。Integration Server は JMS 接続エイリアスを使用して、JMS プロバイダとの間でメッセージを送受信します。

ネイティブ webMethods API を使用した webMethods Broker への接続

JMS プロバイダとして webMethods Broker を使用する場合は、ネイティブ webMethods API を使用するように JMS 接続エイリアスを設定できます。ネイティブ webMethods API を使用すると、JMS プロバイダとして機能する webMethods Broker に直接 Integration Server を接続できます。このように接続すると、接続ファクトリは必要がなくなります。宛先は Broker で直接作成できるため、JNDI プロバイダに保存する必要はありません。このため、すべての JMS 接続エイリアスで、webMethods Broker への接続をネイティブ webMethods API を使用して行うように指定している場合は、JNDI プロバイダを使用する必要はありません。

webMethods Broker は使用するけれども (ネイティブ webMethods API を使用した) ネイティブな接続は行わない場合は、今までどおり JNDI プロバイダを使用する必要があります。

注:
webMethods Broker は廃止されました。その結果、JMS 接続エイリアスでネイティブ webMethods API を使用する設定も廃止されました。

事前定義済みの JMS 接続エイリアス

Integration Server には、事前定義済みの JMS 接続エイリアスがあります。Integration Server を最初に起動したときに、Integration Server によってエイリアスが作成されます。以前のバージョンから移行した Integration Server にエイリアスが存在していた場合、Integration Server には既存のエイリアスが保持されます。つまり、Integration Server では以前のエイリアス定義は上書きされません。

Integration Server には、以下の事前定義済みの JMS 接続エイリアスがあります。

[JMS 接続エイリアス]	説明
DEFAULT_IS_JMS _CONNECTION	事前定義済みの JNDI プロバイダエイリアス DEFAULT_IS_JNDI_PROVIDER を使用して、Universal Messaging への接続を定義する JMS 接続エイリアス。 この JMS 接続エイリアスは、Integration Server を最初に起動したときに Integration Server によって設定されます。 この接続エイリアスは、デフォルトでは無効になっています。
PE_NON TRANSACTIONAL_ ALIAS	事前定義済みの JNDI プロバイダエイリアス DEFAULT_IS_JNDI_PROVIDER を使用して、Universal Messaging への接続を確立する Process Engine の JMS 接続エイリアス。 このエイリアスは、Integration Server を最初に起動したときに Integration Server によって設定されます。 **注:** Process Engine がインストールされていない場合でも、このエイリアスは Integration Server によって作成されます。

注:
以前のバージョンから Integration Server 9.10 以降に移行した場合は、EventBus エイリアスがある場合があります。これは 9.10 より前のバージョンにおいて事前定義された JMS 接続エイリアスでした。この EventBus エイリアスは EDA イベントバスサーバ用であり、事前定義された JNDI プロバイダエイリアス DEFAULT_IS_JNDI_PROVIDER を使用して、Universal Messaging への接続を定義します。

JMS 接続エイリアスの作成

JMS 接続エイリアスを作成するときは、以下の点に留意してください。

- JNDI を使用して管理対象オブジェクト (接続ファクトリおよび宛先) を抽出してから、接続ファクトリを使用して接続を作成できます。JNDI プロバイダを使用する場合は、JMS 接続エイリアスを作成する前に 1 つ以上の JNDI プロバイダエイリアスを設定する必要があります。

  または

  ネイティブ webMethods API を使用すると、webMethods Broker 上で直接接続を作成できます。Broker で宛先を作成できるため、ネイティブ webMethods API を使用する場合は JNDI プロバイダエイリアスを設定する必要はありません。

  webMethods Broker は使用するけれどもネイティブには接続しない場合は、JNDI プロバイダを使用して 1 つ以上の JNDI プロバイダエイリアスを設定する必要があります。

- 各 JMS 接続エイリアスには、トランザクションタイプが関連付けられています。Integration Server 内の特定の機能は、トランザクションに使用されていないセッション内で完了する必要があります。たとえば、Integration Server を使用して大量のメッセージストリームを送受信するには、トランザクションタイプを NO_TRANSACTION に設定した JMS 接続エイリアスを指定する必要があります。

- オンプレミスの Integration Server からクラウドの Universal Messaging サーバへの JMS 接続エイリアスを設定する場合は、custom_wrapper.conf 設定ファイルを変更して `com.softwareag.um.jndi.cf.url.override=true` を含める必要があります。具体的には、custom_wrapper.conf ファイルに以下を含めるように変更してください。

  `wrapper.java.additional.`*n*`=-Dcom.softwareag.um.jndi.cf.url.override=true`

  ここで *n* は次に使用可能な wrapper.java.additional 番号です。

JMS 接続エイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JMS 設定] に移動します。
3. [**JMS 接続エイリアスの作成**] をクリックします。
4. JMS 接続エイリアスの [**全般設定**] を以下のように設定します。

フィールド	指定内容
[接続エイリアス名]	接続エイリアスの名前。各接続エイリアスは特定の JMS プロバイダへの接続ファクトリを表します。
説明	JMS 接続エイリアスの説明。
[トランザクションタイプ]	この JMS 接続エイリアスを使用するセッションがトランザクションに使用されるかどうか。

選択項目	目的
NO_TRANSACTION	この JMS 接続エイリアスを使用するセッションはトランザクションには使用されないことを指定します。
LOCAL_TRANSACTION	この JMS 接続エイリアスを使用するセッションがローカルトランザクションの一部になることを指定します。
XA_TRANSACTION	この JMS 接続エイリアスを使用するセッションが XA トランザクションの一部になることを指定します。

[接続クライアント ID]	この JMS 接続エイリアスによって確立される接続に関連付けられる JMS クライアント識別子。 **注:** 接続クライアント ID は、JMS 接続エイリアスまたは接続ファクトリで指定できます。接続クライアント ID を両方で指定した場合は、接続ファクトリの値が Integration Server によって使用されます。接続クライアント ID の詳細については、320 ページの「接続クライアント ID について」を参照してください。
[ユーザ (オプション)]	接続ファクトリから接続を取得するために必要なユーザ名。 接続を取得するためのユーザ名およびパスワードが JMS プロバイダに必要かどうかの詳細については、JMS プロバイダの製品資料を参照してください。
[パスワード (オプション)]	接続ファクトリから接続を取得するために必要なパスワード。 接続を取得するためのユーザ名およびパスワードが JMS プロバイダに必要かどうかの詳細については、JMS プロバイダの製品資料を参照してください。

5. [**接続の作成方法**] リストから、以下のいずれかを選択して管理対象オブジェクト (接続ファクトリおよび宛先) の検索方法を指定します。

   - JNDI プロバイダを使用する場合は、[**JNDI 検索**] を選択します。

- ネイティブ webMethods API を使用して webMethods Broker 上で直接接続を作成する場合は、[**ネイティブ webMethods API**] を選択します。

6. [**接続の作成方法**] リストで [**JNDI 検索**] を選択した場合は、[**接続プロトコル設定**] の下にある残りのフィールドに次のように指定します。

フィールド	指定内容
[JNDI プロバイダのエイリアス名]	この JMS 接続エイリアスで管理対象オブジェクトの検索に使用する JNDI プロバイダへのエイリアス。JNDI プロバイダエイリアスの作成の詳細については、299 ページの「JNDI プロバイダエイリアスの作成」を参照してください。
[接続ファクトリ検索名]	この JMS 接続エイリアスで指定された JMS プロバイダ接続を作成するために使用する接続ファクトリの検索名。 JMS 接続エイリアスが Universal Messaging に接続する場合は、環境を設定したときに作成した Universal Messaging 接続ファクトリを指定します。 SonicMQ を JMS プロバイダとして使用している場合は、SonicMQ オブジェクト定義が含まれるシリアライズ可能な Java オブジェクトファイルを参照するルックアップ名を指定します。ルックアップ名には .sjo 拡張子を含めてください。
[オンデマンドでの管理対象オブジェクトの作成](Universal Messaging)	Integration Server が管理対象オブジェクトを検索した時に、そのオブジェクトが存在しない場合、Integration Server が JNDI プロバイダに管理対象オブジェクトが作成するかどうか。 オブジェクトの JNDI 検索が失敗したときに Integration Server で宛先または接続ファクトリを作成する場合はチェックボックスをオンにします。オンデマンドでの管理対象オブジェクトの作成の詳細については、329 ページの「管理オブジェクトの作成」を参照してください。 オンデマンドで管理対象オブジェクトを作成するには、JNDI プロバイダが Universal Messaging JNDI プロバイダであり、JMS プロバイダが Universal Messaging であることが必要です。 **注:** Software AG では、実稼働環境でこの機能を使用しないことをお勧めします。つまり、実稼働環境で実行されている Integration Server に対して、[**オンデマンドでの管理対象オブジェクトの作成**] チェックボックスをオフのままにします。
[マスター追跡の有効化](Universal Messaging)	この JMS 接続エイリアスから作成された接続が常に Universal Messaging クラスタのマスター領域サーバに接続するかどうか。この設定は、この JMS 接続エイリアスを使用して作成されたプロデューサ接続とコンシューマ接続に影響します。次のいずれかの手順に従います。

フィールド	指定内容
	■ Integration Server がこの JMS 接続エイリアスを使用してメッセージを送信または受信するときに Integration Server が常に Universal Messaging クラスタのマスター領域サーバに接続することを示すには、[**マスター追跡の有効化**] チェックボックスをオンします。 ■ Integration Server がこの JMS 接続エイリアスを使用してメッセージを送信または受信するときにこの JMS 接続エイリアスのマスターを追跡する動作を無効にするには、[**マスター追跡の有効化**] チェックボックスをオフします。 ソリューションで JMS メッセージの送信のラウンドロビン機能を使用する場合、JMS 接続エイリアスのマスターを追跡する設定を無効にします。 **注:** [**マスター追跡の有効化**] オプションは、Universal Messaging を JMS プロバイダとして使用する JMS 接続エイリアスで使用できます。
[webMethods 接続ファクトリの監視]	Integration Server が接続ファクトリオブジェクトの変更を監視する方法 (変更がある場合)。これは、JMS 接続エイリアスが JNDI サーバの webMethods 接続ファクトリオブジェクトを使用して webMethods Broker に接続する場合にのみ適用されます。 **選択項目** / **目的** **[いいえ]** Integration Server で接続ファクトリを監視しないことを指定します。これがデフォルトです。 **[変更ポーリング (ポーリング間隔を指定する)]** 指定した間隔で変更をポーリングすることで接続ファクトリを監視します。 **[変更ポーリング (ポーリング間隔は webMethods 接続ファクトリにより定義される)]** webMethods 接続ファクトリオブジェクトに指定したリフレッシュ間隔によって決定される間隔で接続ファクトリを監視します。クラスタ接続ファクトリの設定の詳細については、『*Administering webMethods Broker*』および『*webMethods Broker Messaging Programmer's Guide*』を参照してください。 **[変更リスナーの登録]** イベントリスナーを登録することで接続ファクトリを監視します。
[ポーリング間隔 (分)]	ポーリング試行の間隔 (分単位)。ポーリング間隔は正の整数にする必要があります。デフォルト値は 60 分です。

フィールド	指定内容
	注: このフィールドは、[**変更ポーリング(ポーリング間隔を指定する)**]を選択した場合にのみ使用できます。

注:
詳細については、331 ページの「接続ファクトリオブジェクトの変更の監視」を参照してください。

7. [**接続の作成方法**] リストで [**ネイティブwebMethodsAPI**] を選択した場合は、以下の手順に従って、この JMS 接続エイリアスの JMS プロバイダとして使用する Broker Server への接続を設定します。

フィールド	指定内容
[**Broker ホスト**]	Broker Server がインストールされているマシンの名前 (*DNSname*:*port* または *ipaddress*:*port*)。
[**Broker 名**]	Broker Server に定義されている webMethods Broker の名前。デフォルトの名前は「**Broker #1**」です。
[**クライアントグループ**]	Integration Server がJMS クライアントとして機能する場合に属するクライアントグループの名前。指定するクライアントグループは既に Broker Server に存在している必要があります。デフォルトは「IS-JMS」です。
[**Broker リスト**]	(JMS クライアントとして機能する) Integration Server と (JMS プロバイダとして機能する) Broker 間の接続を配置できる Broker Server のカンマ区切りリスト。このリストによって、接続のフェイルオーバーが提供されます。リスト内の最初の Broker Server との接続に失敗すると、リストされている次の Broker Server に対して接続が試行されます。webMethods Broker ごとに以下のフォーマットを使用します。 {webMethods Broker 名]@<ホスト>[:ポート] **注:** ([**メッセージング**] > [**webMethods 設定**] を介して) Broker Server への接続が既に設定されている場合、Integration Server は [接続プロトコル設定] の下にあるフィールドにその Broker Server の情報を移入します。その Broker Server が JMS プロバイダとして機能する場合、情報の編集が不要になることがあります。ただし、[クライアントグループ] フィールドには、Integration Server が JMS クライアントとして機能する場合に属するクライアントグループが指定されていることを確認してください。
[**キーストア**]	この Integration Server のキーストアファイルの完全パス。キーストアファイルは、SSL 認証に必要となるクレデンシャル (秘密鍵/署

フィールド	指定内容
	名付き認証) を含みます。Broker Server が SSL 接続を必要とする場合は、Integration Server クライアントを Broker Server に対して認証するために、このファイル内の情報が使用されます。 Integration Server のキーストアファイルは、Integration Server が常駐するマシンに保存されます。
[キーストアタイプ]	Integration Server のキーストアファイルのタイプ。「**PKCS12**」または「**JKS**」のいずれかになります。
[トラストストア]	この Integration Server クライアントのトラストストアファイルの完全パス。トラストストアファイルは、SSL 認証の署名責任を負う認証局の信用のあるルート認証を含みます。SSL 接続を確立するには、トラストストアファイルから、キーストアに保存された SSL 認証の有効で信用のあるルートにアクセスできる必要があります。 Integration Server のトラストストアファイルは、Integration Server が常駐するマシンに保存されます。
[トラストストアタイプ]	Integration Server のトラストストアファイルのファイルタイプ。「**JKS**」になります。
鍵の署名と	Integration Server と webMethods Broker の間の接続を暗号化するかどうかの指定。

8. [**高度な設定**] で、JMS 接続エイリアスに関する以下の情報を指定します。

フィールド	指定内容
[クラスローダ]	この JMS 接続エイリアスで使用するクラスローダの名前。Integration Server は、JMS 接続エイリアスを使用した特定のアクティビティ (メッセージの送信、メッセージの受信、接続の作成、宛先の作成など) の実行時に、指定したクラスローダを使用します。 デフォルトでは、Integration Server はサーバクラスローダを使用します。ただし、サーバクラスローダの代わりにパッケージのクラスローダを指定することもできます。パッケージのクラスローダはサードパーティの JMS プロバイダを使用するときに役立つことがあります。たとえば、各 JMS プロバイダに必要なサードパーティの jar ファイルを別々のパッケージ、具体的には *Integration Server_directory* /instances/*instance_name*/packages/*packageName*/code/jars ディレクトリに配置することができます。上記の操作を行うと、それぞれの JMS プロバイダに必要な jar ファイル間で競合が発生するのを防ぐことができます。

フィールド	指定内容
[最大 CSQ サイズ (メッセージ)]	この JMS 接続エイリアスのクライアントサイドキューに入れることができるメッセージの最大数。Integration Server では、メッセージの送信時に JMS プロバイダを使用できなかった場合、クライアントサイドキューにメッセージを書き込みます。それぞれの JMS 接続エイリアスには専用のクライアントサイドキューがあります。 クライアントサイドキューで無制限の数のメッセージを格納できるようにする場合は、「-1」を指定します。つまり、最大限度を設定しない場合は「-1」を指定します。 「0」を設定すると、Integration Server はこの JMS 接続エイリアスのクライアントサイドキューにメッセージを書き込みません。
[CSQ を順番に排出]	Integration Server がクライアントサイドキューを排出する際に、Integration Server がクライアントサイドキューにメッセージを入れた順番で JMS プロバイダにメッセージを送信するかどうか。 Integration Server がクライアントサイドキューからメッセージを送信する場合に、Integration Server が元々クライアントサイドキューにメッセージを入れた順番で送信するように設定する場合は、このチェックボックスをオンにします。 **[CSQ を順番に排出]** チェックボックスをオンにした場合、JMS プロバイダへの接続が再確立された後、Integration Server は、クライアントサイドキューを完全に排出するまで、クライアントサイドキューに新しいメッセージを書き込み続けます。**[CSQ を順番に排出]** チェックボックスをオフにした場合、JMS プロバイダへの接続が再確立された後、Integration Server は、クライアントサイドキューを排出する一方で、JMS プロバイダに直接新しいメッセージを送信します。 **注:** また、Integration Server が JMS プロバイダにデリバーするためにクライアントサイドキューから一度に抽出するメッセージの数を指定することもできます。デフォルトでは、Integration Server は一度に 25 個のメッセージを送信します。watt.server.jms.csq.batchProcessingSize プロパティの詳細については、1095ページの「サーバ設定パラメータ」を参照してください。
[一時キューの作成]	Integration Server が JMS プロバイダ上に一時キューを作成して、replyTo 宛先を指定していない要求/応答操作を処理するかどうか。 Integration Server で一時キューを作成する場合は、チェックボックスをオンにします。Integration Server で一時キューを作成しない場合は、チェックボックスをオフにします。

フィールド	指定内容
	[**一時キューで要求応答リスナを有効にする**] チェックボックスを選択する場合は、[**一時キューの作成**] チェックボックスを選択する必要があります。
[一時キューで要求応答リスナを有効にする]	Integration Server がこの JMS 接続エイリアスの一時キューに送信される同期応答を受信するために、1 つの専用の MessageConsumer を作成するかどうかを指定します。このチェックボックスをクリアすると、Integration Server は応答メッセージごとに新しい JMS MessageConsumer を作成します。多くの状況で、応答ごとに 1 つの MessageConsumer を作成しても、パフォーマンスに影響はありません。ただし、多くのスレッドが pub.jms:sendAndWait を同時に呼び出すなどの状況では、予想される応答ごとに MessageConsumer を作成すると、パフォーマンスに影響を与える可能性があります。このチェックボックスを選択すると、Integration Server はこの JMS 接続エイリアスを使用してパブリッシュされる要求への応答を受信するために、専用のコンシューマを作成します。応答の受信に対して専用のリスナを作成する詳細については、321 ページの「応答を受信するための専用リスナの作成」を参照してください。
[Designer で宛先管理の有効化] (Broker および Universal Messaging)	ユーザが JMS トリガーを使用するときに、Designer を使用して、webMethods Broker または 上で宛先を作成、リストおよび変更できるかどうか。 Designer ユーザが、この JMS 接続エイリアスを使用する JMS トリガーを使用して、宛先を作成、リストおよび変更できるようにする場合は、このチェックボックスをオンにします。 Software AG では、実稼働環境では Designer ユーザに宛先を管理させないことをお勧めします。 Designer を使用して宛先を管理する方法の詳細については、318 ページの「JMS 接続エイリアスおよび Designer による宛先管理の許可」を参照してください。
[トリガーごとに新規接続を作成する]	Integration Server で JMS プロバイダへの接続を JMS トリガーごとに別々に作成するかどうか。 この JMS 接続エイリアスを使用する JMS トリガーごとに Integration Server で別々の接続を作成する場合は、このチェックボックスをオンにします。 この JMS 接続エイリアスを使用する同時 JMS トリガーで複数の JMS プロバイダ接続を使用する場合は、トリガーごとに別々の接続を作成するようにエイリアスを設定する必要があります。 詳細については、319 ページの「JMS 接続エイリアスの複数接続の許可」を参照してください。

9. [**プロデューサキャッシュ**] で次のように指定して、JMS Session オブジェクトおよび MessageProducer オブジェクトのキャッシュ用プールを設定します。プロデューサキャッシュの詳細については、322 ページの「JMS メッセージの送信のためのプロデューサキャッシュの設定」を参照してください。

フィールド	指定内容
[キャッシュモード]	この接続エイリアスの JMS Session オブジェクトおよび MessageProducer オブジェクトのキャッシュを有効化するかどうか。 **選択項目** / **目的** **[無効]**: Integration Server が JMS Session オブジェクトまたは MessageProducer オブジェクトをキャッシュしないことを示します。 **[宛先ごとに有効]**: JMS Session オブジェクトおよび MessageProducer オブジェクトのキャッシュを有効にします。 トランザクションに使用されない JMS 接続エイリアスの場合、Integration Server は Session オブジェクトと MessageProducer オブジェクトをキャッシュします。 トランザクションに使用される JMS 接続エイリアスの場合、Integration Server は MessageProducer オブジェクトをキャッシュします。 **注:** JMS プロバイダが Universal Messaging または WebSphere MQ 7.5 である場合、Integration Server はトランザクションに使用される JMS 接続エイリアスのセッションキャッシュをサポートします。Integration Server のこのバージョンでサポートされている Universal Messaging のバージョンの詳細については、336 ページの「サポートされている JMS プロバイダ」を参照してください。
[最小プールサイズ (宛先の指定なし)]	デフォルトセッションプールのエントリの最小数。デフォルトは 1 です。
[最大プールサイズ (宛先の指定なし)]	デフォルトセッションプールのエントリの最大数。デフォルトは 30 です。
[宛先ごとの最小プールサイズ]	各宛先固有プールのエントリの最小数。

フィールド	指定内容
[宛先ごとの最大プールサイズ]	各宛先固有プールのエントリの最大数。値 0 (または空白) は、Integration Server が JMS 接続エイリアスに関連付けられている任意の宛先に対して個別のプールを作成しないことを示します。
[宛先ルックアップ名リスト]	Integration Server が個別のプールを作成する宛先の検索名のセミコロン区切りのリスト。 **注:** このフィールドは、JMS 接続エイリアスで JMS プロバイダへの接続を作成するために JNDI 検索を指定した場合にのみ表示されます。
[キューリスト]	Integration Server が個別のプールを作成するキューのセミコロン区切りのリスト。 **注:** このフィールドは、JMS 接続エイリアスで webMethods Broker への接続を作成するためにネイティブ webMethods API を指定した場合にのみ表示されます。
[トピックリスト]	Integration Server が個別のプールを作成するトピックのセミコロン区切りのリスト。 **注:** このフィールドは、JMS 接続エイリアスで webMethods Broker への接続を作成するためにネイティブ webMethods API を指定した場合にのみ表示されます。
[アイドルタイムアウト]	Integration Server が非アクティブのプールエントリを削除するまでのミリ秒数。このタイムアウト値は、デフォルトのセッションプールおよび宛先固有プールに適用されます。 値 0 はプールエントリが期限切れにならないことを示します。値 -1 は、Integration Server が watt.server.jms.producer.pooledSession.timeout パラメータで定義されているシステムデフォルトを使用することを示します。 このデフォルトは 60000 ミリ秒です。

10. [**プロデューサ再試行**] で以下のフィールドを指定して、この JMS 接続エイリアスを使用して JMS プロバイダにメッセージを送信する pub.jms:send サービスの自動再試行を設定します。サービスの自動再試行を設定する方法の詳細については、324 ページの「pub.jms:send サービスを使用して JMS メッセージを送信する場合の自動再試行の設定」を参照してください。

フィールド	指定内容
[最大再試行回数]	Integration Server が一時的なエラーにより失敗した pub.jms:send サービスを自動的に再試行する最大回数です。値 0 は、この JMS 接続エイリアスでは自動再試行が無効化されていることを示します。デフォルトは 0 です。
[再試行間隔 (ミリ秒)]	再試行の間に Integration Server が待機するミリ秒数です。デフォルトは 1000 ミリ秒 (1 秒) です。

注:
JMS 接続エイリアスがトランザクションに使用されるか、マルチ送信保証付きポリシーが適用されている接続ファクトリを使用する場合、Integration Server は、プロデューサ再試行値を無視します。

11. [**拡張ロギング**] で以下の内容を指定し、このメッセージング接続エイリアスを使用する JMS メッセージの送受信に関する追加ログを設定します。

フィールド	指定内容
[ログタイプ]	この JMS 接続エイリアスを使用してメッセージを送受信するメッセージプロデューサやコンシューマに対し拡張ロギングが有効になっている場合に、Integration Server がログメッセージを書き込む先。 ■ [**サーバログ**]。拡張ロギングメッセージをサーバログに書き込みます。サーバログを宛先に指定する場合、0168 メッセージング (拡張ロギング) サーバログ機能のログレベルを Info 以上に上げてください。 ■ [**メッセージング監査ログ**]。拡張ロギングメッセージをメッセージング監査ログに書き込みます。 上記のいずれかのオプションのみを選択できます。Integration Server は拡張ロギングメッセージをサーバログとメッセージング監査ログの両方に書き込むことはできません。
[プロデューサメッセージ ID の追跡を有効化]	選択した場合、メッセージプロデューサがこの JMS 接続エイリアスを使用して [**プロデューサメッセージ ID 追跡: インクルードされた宛先**] の宛先にメッセージを送信するとき、Integration Server が追加ログメッセージを書き込むことを示します。
[プロデューサメッセージID追跡: インクルードされた宛先]	Integration Server がメッセージを宛先に送信するときに追加ログを記録する宛先名。セミコロン (;) を使用して、各宛先名を区切ります。この JMS 接続エイリアスがメッセージを送信するすべての宛先で Integration Server による拡張ロギングを実行したい場合、このフィールドを空白のままにします。
[コンシューマメッセージ ID の追跡を有効化]	選択した場合、Integration Server が、この JMS 接続エイリアスを使用してメッセージを受信するメッセージングコンシューマ (トリガー) の追加ログメッセージを書き込むことを示します。Integration Server は

フィールド	指定内容
	[**コンシューマメッセージ ID 追跡: インクルードされたトリガー**] に示された JMS トリガーの追加ログメッセージを書き込みます。
[**コンシューマメッセージ ID 追跡: インクルードされたトリガー**]	Integration Server がトリガー処理時に追加ログを記録する対象の JMS トリガーの完全修飾名。セミコロン (;) を使用して、各トリガー名を区切ります。この JMS 接続エイリアスを使用してメッセージを受信するすべての JMS トリガーで Integration Server による拡張ロギングを実行したい場合、このフィールドを空白のままにします。

JMS メッセージの送受信の拡張ロギングの使用方法については、357 ページの「メッセージングの拡張ロギングの使用」を参照してください。

12. [**変更内容の保存**] をクリックします。

13. JMS 接続エイリアスを有効にします。

JMS 接続エイリアスおよび Designer による宛先管理の許可

Designer ユーザが JMS トリガーを操作するときに宛先および継続的サブスクライバを管理できるように、JMS 接続エイリアスを設定できます。JMS 接続エイリアスが宛先を管理するように設定されている場合 ([**Designer による宛先管理を有効にする**] チェックボックスがオン)、Designer を使用して webMethods Broker または Software AG Universal Messaging で宛先および継続的サブスクライバを作成および変更できます。

注:
Broker または Universal Messaging で Designer を使用して宛先を管理する機能は、設計時の機能です。実稼動環境では、この機能を無効にする必要があります。

Broker で宛先を管理するには、以下を満たしている必要があります。

- JMS 接続エイリアスが、Broker を JMS プロバイダとして使用している必要があります。
- Broker は Broker バージョン 7.1 以上である必要があります。
- Integration Server にインストールされている次の 3 つの Broker jar ファイルのバージョンが 8.0 SP1 以上である必要があります。
  - *Software AG_directory* /common/lib/wm-brokerclient.jar
  - *Software AG_directory* /common/lib/wm-jmsclient.jar
  - *Software AG_directory* /common/lib/wm-jmsnaming.jar

Universal Messaging で宛先を管理するには、以下を満たしている必要があります。

- JMS 接続エイリアスが、Universal Messaging を JMS プロバイダとして使用している必要があります。

- Universal Messaging のバージョンは Integration Server のこのバージョンでサポートされている必要があります。サポートされているバージョンの詳細については、336 ページの「サポートされている JMS プロバイダ」を参照してください。

Designer を使用して Broker または Universal Messaging 上で宛先を変更する方法の詳細については、『*webMethods Service Development Help*』を参照してください。

JMS 接続エイリアスの複数接続の許可

JMS 接続エイリアスを使用する JMS トリガーごとに Integration Server で個別の JMS プロバイダ接続を作成するように、エイリアスを設定できます。個々のトリガーに対して別々の接続を作成すると、特に多数のトリガーで大量のメッセージを処理する場合にパフォーマンスを向上させることができます。

デフォルトでは、エイリアスは単一の JMS プロバイダ接続を作成し、エイリアスを使用する個々の JMS トリガーで同じ接続を共有します。Integration Server は、JMS 接続エイリアスが指定されている pub.jms* サービスの実行時に、この同じ接続を使用して JMS メッセージを送信します。

JMS 接続エイリアスの設定時に [**トリガーごとに新規接続を作成する**] チェックボックスをオンにした場合、Integration Server は、エイリアスを使用する JMS トリガーごとに新しい JMS プロバイダ接続を作成します。ここで作成される接続は、Integration Server が JMS メッセージの送信に使用する接続とは別に作成されます。そのため、JMS 接続エイリアスが 3 つの JMS トリガーに関連付けられている場合、そのエイリアスに関連付けられている接続の数は合計で 4 つになります。

エイリアスでトリガーごとに別々の接続を作成する場合、複数の JMS プロバイダ接続を取得するように関連する同時 JMS トリガーを設定して、トリガーのスループットを著しく向上させることができます。ただし、現在のスループットに関係なく、トリガーによって使用される個々の接続には専用の Integration Server スレッドが必要であることに注意してください。

注:
[**トリガーごとに新規接続を作成する**] チェックボックスをオンにすると、関連付けられている JMS トリガーの [**ローカルパブリッシュを無視**] 機能が動作しなくなります。トリガーにローカルパブリッシュメッセージを無視させるには、パブリッシャーとサブスクライバとで同じ接続を共有する必要があります。JMS 接続エイリアスでトリガーごとに新しい接続を作成すると、パブリッシャーとサブスクライバとで同じ接続を共有しなくなります。

Integration Server では、サポートされている JMS プロバイダについて、単一の JMS 接続エイリアスで複数の接続を作成および使用することがサポートされています。サポートされている JMS プロバイダの一覧については、336 ページの「サポートされている JMS プロバイダ」を参照してください。

注:
Integration Server によって複数の接続が作成された場合、Integration Server では各接続に同じクライアント ID を使用します。このことは webMethods Broker では許可されますが、一部の JMS プロバイダでは、同じクライアント ID での複数の接続を使用することはサポートされていないか、サポートされるようにするには追加設定が必要になります。このことは、トピックや継続的サブスクライバーを使用する場合に特に該当します。複数の接続を使用す

るように JMS 接続エイリアスまたは JMS トリガーを設定する前に、JMS プロバイダのマニュアルを確認してください。

JMS プロバイダとして webMethods Broker を使用する場合に、単一の JMS 接続エイリアスで複数の接続を使用するには、以下を満たしている必要があります。

- webMethods Broker は webMethods Broker バージョン 7.1 以上である必要があります。
- Integration Server にインストールされている次の 3 つの webMethods Broker jar ファイルのバージョンが 8.0 SP1 以上である必要があります。
  - *Software AG_directory* /common/lib/wm-brokerclient.jar
  - *Software AG_directory* /common/lib/wm-jmsclient.jar
  - *Software AG_directory* /common/lib/wm-jmsnaming.jar

JMS トリガーの設定の詳細については、Software AG Designer の『*webMethods Service Development Help*』を参照してください。

接続クライアント ID について

接続クライアント ID は、JMS 接続エイリアスによって確立される接続に関連付けられる JMS クライアント識別子です。JMS 接続エイリアスから作成される接続に対して Integration Server がどのような接続クライアント ID を使用するかは、以下の 1 つ以上の条件によって異なります。

- JMS 接続エイリアスの [**接続クライアント ID**] フィールドの値
- JMS 接続エイリアスによって使用される接続ファクトリで指定されている接続クライアント ID
- JMS 接続エイリアスが JMS トリガーごとに新しい接続を作成するように設定されているかどうかつまり、[**トリガーごとに新規接続を作成する**] チェックボックスがオンかどうか

上記の情報を使用して、Integration Server では、接続クライアント ID を以下のように決定します。

- 接続クライアント ID が JMS 接続エイリアスによってのみ指定される場合、Integration Server では、エイリアスから作成される接続に対して、この値を使用します。webMethods Broker にネイティブ接続する ([**接続の作成方法**] リストが [**ネイティブwebMethodsAPI**] に設定されている) 場合、Integration Server では、JMS 接続エイリアスからの接続クライアント ID を常に使用します。
- 接続クライアント ID が接続ファクトリによってのみ指定される場合、Integration Server では、エイリアスから作成される接続に対して、この値を使用します。
- 接続クライアント ID が JMS 接続エイリアスと接続ファクトリによって指定される場合、Integration Server では接続ファクトリの値を使用します。このことは、webMethods Broker を含む、すべての JMS プロバイダを使用する場合に該当します。

- [**トリガーごとに新規接続を作成する**] チェックボックスがオンではない場合、JMS 接続エイリアスを使用する各 JMS トリガーでは、同じ接続を使用します。各接続の接続クライアント ID は同じになります。
- [**トリガーごとに新規接続を作成する**] チェックボックスがオンの場合、JMS 接続エイリアスを使用する各 JMS トリガーでは、JMS プロバイダへの独自の接続を作成します。
  - Integration Server によって接続ファクトリの接続クライアント ID が使用される場合、1 つの JMS トリガーに対する各接続の接続クライアント ID は同じになります。
  - Integration Server によって JMS 接続エイリアスの接続クライアント ID が使用される場合、1 つの JMS トリガーに対する各接続は、その JMS トリガーについて一意になります。接続クライアント ID は、JMS 接続エイリアスの [**接続クライアント ID**] フィールドの値と JMS トリガーの完全修飾名で構成されます。

注:
負荷分散方式でメッセージを受信するには、それぞれの JMS トリガーで同じ接続クライアント ID を使用して webMethods Broker に接続する必要があります。[**トリガーごとに新規接続を作成する**] オプションによって、接続クライアント ID が変更される場合があるため、このオプションの使用方法は Integration Server グループのすべての Integration Server で一貫している必要があります。

応答を受信するための専用リスナの作成

Integration Server には、パブリッシュされる要求への応答を受信するための専用のコンシューマを作成する機能があります。この機能は、JMS メッセージング接続エイリアスごとに設定します。

pub.jms:sendAndWait サービスが同期要求応答を実行すると、Integration Server は、JMS プロバイダに要求メッセージを送信し、応答を待機します。デフォルトでは、Integration Server は応答メッセージごとに JMS MessageConsumer を作成します。多くの状況で、応答ごとに 1 つの MessageConsumer を作成しても、パフォーマンスに影響はありません。ただし、多くのスレッドが pub.jms:sendAndWait を同時に呼び出すなどの状況では、予想される応答ごとに MessageConsumer を作成すると、パフォーマンスに影響を与える可能性があります。

これに対応するために、Integration Server に [**一時キューで要求応答リスナを有効にする**] というオプションがあります。これを選択すると、JMS 接続エイリアスに送信された同期応答の受信のために、1 つの専用 MessageConsumer が作成されます。Integration Server が 1 つのコンシューマを使用して、特定の JMS 接続エイリアスに送信された要求のすべての同期応答を取得する場合に pub.jms:sendAndWait サービスを実行すると、Integration Server は JMS メッセージのプロパティとして使用される新しい *wm_tag* フィールド *JMSMessage/properties/wm_tag* に一意の値を割り当てます。pub.jms:reply を使用して *wm_tag* が生成された JMS メッセージ要求に応答する場合、応答する Integration Server は *wm_tag* の値を応答メッセージの *JMSMessage/header/JMSCorrelationID* フィールドにマッピングします。

専用のコンシューマを使用して特定の JMS エイリアスを使用するすべての要求に対する応答を取得するには、次の条件を満たす必要があります。

- 要求の送信に使用される JMS 接続エイリアスは次のように設定されている。
  - [**一時キューの作成**] チェックボックスが選択されている。

  - [**一時キューで要求応答リスナを有効にする**] チェックボックスが選択されている。これは JMS 接続エイリアスの新しいオプションです。

- pub.jms:sendAndWait の呼び出しは
  - 同期である (*async* 入力パラメータが false に設定されている)。
  - *destinationNameReplyTo* 入力パラメータの値が指定されていない。

JMS メッセージの送信のためのプロデューサキャッシュの設定

JMS メッセージを送信すると、Integration Server はメッセージごとに新しい JMS Session オブジェクトおよび JMS MessageProducer オブジェクトを作成して閉じます。これにより、一部の JMS プロバイダにオーバーヘッドが発生する場合があります。JMS メッセージを送信するときのパフォーマンスを向上させるには、プロデューサ側でプーリングを設定します。JMS 接続エイリアスごとに、Integration Server は以下を作成できます。

- JMS Session オブジェクトを含むデフォルトのセッションプール。デフォルトのセッションプールが JMS 接続エイリアスに対して定義されている場合、Integration Server は、JMS メッセージごとに JMS Session を開いたり閉じたりするのではなく、JMS メッセージを送信するためにオープンな JMS Session のプールを使用します。Integration Server は、独自のプールが存在しない宛先にメッセージを送信する場合にのみ、デフォルトのセッションプールを使用します。デフォルトのセッションプールを使用する場合、Integration Server は、JMS メッセージを送信するたびに新しい MessageProducer を作成します。
- JMS メッセージを特定の宛先に送信するための宛先固有プールです。Integration Server は、指定された宛先ごとにプールを作成します。宛先固有プールの構成は JMS 接続エイリアスのトランザクションタイプによって異なります。
  - トランザクションに使用されない JMS 接続エイリアスの場合、宛先固有プールのエントリは Session オブジェクトおよび MessageProducer オブジェクトで構成されています。指定された宛先の 1 つに JMS メッセージを送信する場合、Integration Server は、JMS メッセージごとに JMS Session オブジェクトおよび MessageProducer オブジェクトを作成したり閉じたりするのではなく、プールの Session オブジェクトおよび MessageProducer オブジェクトを使用します。
  - トランザクションに使用される JMS 接続エイリアスの場合、宛先固有プールのエントリには Session オブジェクトが含まれます。指定された宛先の 1 つに JMS メッセージを送信する場合、Integration Server は、JMS メッセージごとに Session オブジェクトを作成したり閉じたりするのではなく、プールの Session オブジェクトを使用します。Integration Server は JMS メッセージを送信するごとに、新しい MessageProducer を作成します。

    注:
    JMS プロバイダが Universal Messaging または WebSphere MQ 7.5 である場合、Integration Server はトランザクションに使用される JMS 接続エイリアスのセッションキャッシュをサポートします。Integration Server のこのバージョンでサポートされている Universal Messaging のバージョンの詳細については、336 ページの「サポートされている JMS プロバイダ」を参照してください。

デフォルトのセッションプールおよびすべての宛先プールに最小サイズと最大サイズを指定できます。また、Integration Server が特定のプールを作成する宛先を指定することもできます。JMS

プロバイダに接続するために、JMS 接続エイリアスで接続ファクトリオブジェクトの使用を指定している場合は、単一の宛先リストを指定します。webMethods Broker に接続するために JMS 接続エイリアスでネイティブ webMethods API が指定されている場合は、Integration Server で宛先プールを作成するキューおよびトピックリストに個別のリストを指定する必要があります。

Integration Server が、トランザクションに使用される接続エイリアスおよびトランザクションに使用されない接続エイリアスに指定される情報に基づいて、どのようにセッションプールおよび宛先プールを作成するかを説明する次の例について考えます。

トランザクションに使用されない JMS 接続エイリアス用のセッションプールおよび宛先プールを作成する例

トランザクションに使用されない、「myAlias」という名前の JMS 接続エイリアスが、ネイティブ webMethods API を使用して webMethods Broker に接続し、フィールドが次の表に示したように設定されているとします。

フィールド	値
[最小プールサイズ]	1
[最大プールサイズ]	10
[宛先ごとの最小プールサイズ]	1
[宛先ごとの最大プールサイズ]	5
[キューリスト]	myQueue1; myQueue2
[トピックリスト]	myTopic
[アイドルタイムアウト]	70000

上記の情報を使用して、Integration Server は最小サイズが 1 で最大サイズが 10 のデフォルトのセッションプールを作成します。このプールには、JMS Session オブジェクトのみが含まれます。Integration Server は、独自のプールが存在しない宛先にメッセージを送信する場合に、このプールのエントリを使用します。

また、Integration Server は myQueue1 と myQueue2 の各キューに 1 つずつ、およびトピック myTopic に 1 つの合計 3 つの宛先プールを作成します。これらの各プールの最大サイズは、5 プールエントリです。myQueue1、myQueue2 または myTopic の宛先に送信されるメッセージでは、その宛先用に作成されたプールのエントリ (Session オブジェクトおよび MessageProducer オブジェクト) が使用されます。他の宛先に送信されるメッセージでは、デフォルトのセッションプールの Session が使用されます。

デフォルトプールまたは宛先固有プールのエントリは、エントリの非アクティブ状態が 70000 ミリ秒 (70 秒) を超えると期限切れになります。

トランザクションに使用される JMS 接続エイリアス用のセッションプールおよび宛先プールを作成する例

トランザクションに使用される、「myAlias1」という名前の JMS 接続エイリアスが、JNDI を使用して WebSphere MQ 7.5 に接続し、フィールドが次の表に示したように設定されているとします。

フィールド	値
[最小プールサイズ]	1
[最大プールサイズ]	15
[宛先ごとの最小プールサイズ]	1
[宛先ごとの最大プールサイズ]	10
[宛先ルックアップ名リスト]	myQueue1; myQueue2; myTopic1
[アイドルタイムアウト]	80000

上記の情報を使用して、Integration Server は最小サイズが 1 で最大サイズが 15 のデフォルトのセッションプールを作成します。このプールには、JMS Session オブジェクトのみが含まれます。Integration Server は、独自のプールが存在しない宛先にメッセージを送信する場合に、このプールのエントリを使用します。

また、Integration Server は myQueue1 と myQueue2 の各キューに 1 つずつ、およびトピック myTopic1 に 1 つの合計 3 つの宛先プールを作成します。これらの各プールの最大サイズは、10 プールエントリです。myQueue1、myQueue2 または myTopic1 の宛先に送信されるメッセージでは、その宛先用に作成されたプールのエントリ (Session オブジェクト) が使用されます。他の宛先に送信されるメッセージでは、デフォルトのセッションプールの Session が使用されます。

デフォルトプールまたは宛先固有プールのエントリは、エントリの非アクティブ状態が 80000 ミリ秒 (80 秒) を超えると期限切れになります。

pub.jms:send サービスを使用して JMS メッセージを送信する場合の自動再試行の設定

JMS 接続エイリアスを設定して、JMS 接続エイリアスを使用している pub.jms:send サービスが一時的なエラーで失敗した場合に Integration Server により自動的に再試行されるようにできます。JMS プロバイダにメッセージを送信するための特定の JMS 接続エイリアスを使用している pub.jms:send サービスのインスタンスの自動再試行を設定するには、エイリアスに対して以下の項目を指定します。

- 再試行の最大回数。[**最大再試行回数**] フィールドにより、Integration Server が特定の pub.jms:send サービスを再試行する最大回数が決まります。[**最大試行回数**] が 0 である場合、JMS 接続エイリアスの自動再試行が無効化されていることを示します。
- 試行間の間隔。[**再試行間隔**] フィールドにより、Integration Server が再試行間に待機するミリ秒数が決まります。デフォルトの間隔は 1000 ミリ秒 (1 秒) です。

JMS 接続エイリアスを使用している pub.jms:send サービスが、一時的なエラーの後に再試行されるには、JMS 接続エイリアスは以下の条件も満たしている必要があります。

- JMS 接続エイリアスが有効化されている。
- JMS 接続エイリアスのトランザクションタイプが NO_TRANSACTION である。Integration Server は、トランザクションの一部として実行される pub.jms:send サービスを再試行しません。
- JMS 接続エイリアスで Broker が JMS プロバイダとして指定されている場合、JMS 接続エイリアスはマルチ送信保証付きポリシーが適用されているクラスタ接続ファクトリを使用しないこと。

次の表に、JMS 接続エイリアスが再試行用に設定され、pub.jms:send サービスが一時的なエラーのために失敗した場合に、Integration Server が使用する再試行プロセスを示します。

ステージ	説明
1	pub.jms:send サービスの実行が一時的なエラーにより失敗します。
2	Integration Server は再試行間隔の時間を待機します。 **注:** 再試行の待機中に Integration Server がシャットダウンを開始した場合、Integration Server は待機時間を中断し、サービスを再試行します。 **注:** Integration Server の再試行の待機中に JMS 接続エイリアスが無効化された場合、Integration Server は再試行プロセスを中断します。JMS 接続エイリアスのクライアントサイドキューが有効化されているかどうかにより、JMS メッセージをクライアントキューに書き込むか、pub.jms:send サービスの失敗の原因となった元の例外をスローするかが決まります。詳細については、後続のステージ 5 を参照してください。
3	Integration Server は pub.jms:send サービスを再実行します。
4	一時的なエラーにより再度 pub.jms:send サービスが失敗した場合、次のいずれかの状態となるまで Integration Server は手順 2 および 3 を繰り返します。 ■ pub.jms:send サービスが正常に実行される。 ■ 再試行の最大回数に達したため再試行に失敗する。
5	再試行に失敗した場合、JMS 接続エイリアスがクライアントサイドキューを使用しているかどうかにより、Integration Server は次のいずれかの動作を行います。 ■ クライアントサイドキューが使用されている場合、Integration Server は pub.jms:send サービスにより作成されたメッセージをクライアントサイドキューに書き込みます。

ステージ	説明
	■ クライアントサイドキューが使用されていない場合、Integration Server は JMS プロバイダによってスローされた元の例外をスローします。 **注:** JMS 接続エイリアスで pub.jms:send サービスに対して [**最大 CSQ サイズ**] 値が 0 (ゼロ) より大きく、[*useCSQ*] 入力パラメータが「true」に設定されている場合、クライアントサイドキューは使用されています。

Software AG Universal Messaging が JMS プロバイダである場合の pub.jms:send サービスの再試行について

JMS 接続エイリアスが、一時的なエラーによって pub.jms:send サービスを再試行するように設定されており、Universal Messaging が JMS プロバイダである場合、Integration Server は Universal Messaging 接続ファクトリのローカルインスタンスの一部を変更して、Integration Server により例外がスローされることを防止する、または少なくとも遅らせる場合があります。Integration Server は変更を加えて、Integration Server が Universal Messaging クラスタへの接続を失った場合に Integration Server が例外を直ちにスローしないようにします。代わりに、Universal Messaging が一定期間例外を抑止します。この期間内に、Universal Messaging はクラスタクォーラムの復元を試みます。同時に、Integration Server は Universal Messaging への接続の再確立を試みます。この遅延の間、Integration Server には例外が通知されず、JMS 接続エイリアスは停止しません。ただし、JMS 接続エイリアスを使用する JMS トリガーは、いかなるメッセージも受信しません。さらに、メッセージを送信するために JMS 接続エイリアスを使用中のすべての pub.jms:send サービスが、JMS 接続エイリアスに設定されている再試行間隔および再試行回数に基づいて再試行されます。Universal Messaging がクラスタクォーラムを復元できない場合、Integration Server は例外をスローします。この時点で、JMS 接続エイリアスを使用しているすべての JMS トリガーは停止し、JMS 接続エイリアスを使用して JMS メッセージを送信するすべてのサービスは、直ちに例外をスローします。Integration Server は、次に Universal Messaging への再接続を試みます。

Integration Server は Universal Messaging 接続ファクトリの MaxReconAttempts プロパティが「-1」に設定されている場合、この接続ファクトリに変更を加えます(値 -1 はデフォルトであり、MaxReconAttempts 値が Universal Messaging で変更されていないことを示します)。Integration Server は、Universal Messaging 接続ファクトリのローカルインスタンスに次の変更を加えます。

- ConxExceptionOnFailure を「true」に設定します。
- MaxReconAttempts を「35」に設定します。
- ReconnectInterval を「2000」ミリ秒に設定します。

Integration Server は、Universal Messaging 接続ファクトリのローカルインスタンスのみに変更を加えます。ConnectionFactory は JNDI プロバイダ上で変更されません。これは、ConnectionFactory を使用している他のクライアントは影響を受けないことを意味します。

Integration Server が Universal Messaging 接続ファクトリのローカルインスタンスに変更を加えることを防止するには、Universal Messaging Enterprise Manager を使用して MaxReconAttempts プロパティに「-1」より大きい値を設定します。

JMS 接続エイリアスの編集

JMS 接続エイリアスを作成した後で、作成したエイリアスまたはデフォルトエイリアスのプロパティの変更が必要になることがあります。たとえば、クライアントサイドキューに入れることができるメッセージの最大数を減らして、クライアントサイドキューが占有できるメモリ容量を減少させることができます。エイリアス名を除く、JMS 接続エイリアスのすべてのプロパティを編集できます。

JMS 接続エイリアスを編集する場合は、以下の情報に留意してください。

- JMS 接続エイリアスは、編集する前に無効にする必要があります。
- Integration Server は継続的サブスクライバ名の最初の部分に接続クライアント ID を使用するため、接続クライアント ID を変更すると、エイリアスを使用するトリガーの継続的サブスクライバが作成されます。名前の一部として古い接続クライアント ID を使用する継続的サブスクライバは Universal Messaging から自動的には削除されません。古い継続的サブスクライバはメッセージを含んでいることや、メッセージの受信を続行することがあります。

JMS 接続エイリアスを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JMS 設定] に移動します。
3. [JMS 接続エイリアスの定義] リストから、編集する JMS 接続エイリアスを選択します。Integration Server Administrator によって、接続エイリアスの詳細が表示されます。
4. [**JMS 接続エイリアスの編集**] をクリックします。
5. 接続エイリアスのプロパティを編集します。フィールドの詳細については、299 ページの「JNDI プロバイダエイリアスの作成」を参照してください。[**接続エイリアス名**] フィールドは変更できません。
6. [**変更内容の保存**] をクリックします。

JMS 接続エイリアスの有効化および無効化

JMS 接続エイリアスを有効にすると、Integration Server でエイリアスを使用して、サービスおよび JMS トリガーの代わりに接続の取得、メッセージの送信およびメッセージの受信を実行できるようになります。接続エイリアスを無効にすると、Integration Server によって、エイリアスを使用するすべての JMS トリガーが一時停止されます。また、メッセージの送受信に、無効にした JMS 接続エイリアスを使用しているすべてのサービスが、エラーで終了します。

JMS 接続エイリアスを有効または無効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**メッセージング**] > [JMS 設定] に移動します。

3. [JMS 接続エイリアスの定義] リストの [有効] 列で以下のいずれかを行います。

   - エイリアスが無効になっており、このエイリアスを有効にする場合は、[**いいえ**] をクリックします。

     Integration Server でエイリアスを有効にできない場合は、Integration Server によってエイリアスの下に有効にできない理由を示すメッセージが表示されます。

   - エイリアスが有効になっており、このエイリアスを無効にする場合は、[**はい**] をクリックします。

JMS 接続エイリアスの削除

JMS 接続エイリアスを削除する前に、以下の点について確認します。

- JMS 接続エイリアスは無効になっている。
- JMS 接続エイリアスに依存しているサービスまたは JMS トリガーはない。JMS トリガーで JMS 接続エイリアスを使用していると、Integration Server によって JMS 接続エイリアスの削除が阻止されます。

JMS 接続エイリアスを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**メッセージング**] > [JMS 設定] に移動します。

3. [JMS 接続エイリアスの定義] リストで、エイリアスがまだ無効になっていない場合はエイリアスを無効にします。

4. 削除するエイリアスを検索し、[削除] 列の ✕ をクリックします。Integration Server によって、アクションの確認を求めるダイアログボックスが表示されます。[**OK**] をクリックして、JMS 接続エイリアスの削除を確認します。

接続監視期間の指定

Integration Server は、アクティブな JMS プロバイダ接続の状態を定期的にチェックします。watt.server.jms.connection.monitorPeriod プロパティを使用すると、Integration Server が接続状態をチェックする頻度を設定できます。デフォルトは 45 秒です。

Integration Server と JMS プロバイダ間の接続に失敗すると、Integration Server は 20 秒後に自動的に接続の再確立を試行します。watt.server.jms.connection.retryPeriod プロパティの値を変更すると、Integration Server が接続の再確立を試行する間隔を設定できます。

サーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

失敗した接続の再試行間隔の指定

Integration Server と JMS プロバイダ間の接続に失敗すると、Integration Server は 20 秒後に自動的に接続の再確立を試行します。watt.server.jms.connection.retryPeriod プロパティの値を変更すると、Integration Server が接続の再確立を試行する間隔を設定できます。このプロパティの詳細については、1136 ページの「watt.server.」を参照してください。

キープアライブ間隔の指定

Integration Server は JMS プロバイダに対して定期的に ping を実行して、Integration Server と JMS プロバイダ間の接続をアクティブに保ちます。ping はキープアライブ要求として機能します。デフォルトでは、Integration Server は 300 秒ごとに JMS プロバイダに対して ping を実行します。watt.server.jms.connection.pingPeriod プロパティの値を変更することで、Integration Server が JMS プロバイダに対して ping を実行する頻度を設定できます。このプロパティの詳細については、1136 ページの「watt.server.」を参照してください。

管理オブジェクトの作成

選択した JMS プロバイダによって、JNDI ネームスペースに管理オブジェクトを作成および設定するツールが提供されます。ほとんどの JMS プロバイダの場合、Integration Server を使用して管理オブジェクトを作成および設定することはできません。管理オブジェクトの使用の詳細については、選択した JMS プロバイダの製品資料を参照してください。

ただし、Universal Messaging が JMS プロバイダおよび JNDI プロバイダである場合、Integration Server では JNDI ネームスペースに管理オブジェクトを作成できます。Integration Server では、管理オブジェクトの検索が失敗すると、オブジェクトが自動的に作成されます。次に例を示します。

- 存在しない接続ファクトリを使用して JMS 接続エイリアスが Universal Messaging に接続しようとすると、Integration Server では接続ファクトリを作成し、Universal Messaging の JNDI ネームスペースに追加します。Integration Server は JNDI 設定から 接続 URL をコピーします。
- pub.jms* サービスによってトピックまたはキューがメッセージ宛先として指定され、その宛先が存在しない場合、Integration Server では Universal Messaging にトピックまたはキューを作成し、JNDI ネームスペースに追加します。
- 存在しないトピックまたはキューに JMS トリガーがサブスクライブした場合、Integration Server では Universal Messaging にトピックまたはキューを作成し、JNDI ネームスペースに追加します。

見つからない管理オブジェクトが Integration Server で自動的に作成されるかどうかは、JMS 接続エイリアスにより [**オンデマンドで管理オブジェクトを作成**] チェックボックスを使用して設定します。

JMS 接続エイリアスが宛先と接続ファクトリをオンデマンドで作成するよう設定した場合、接続ファクトリまたは宛先を作成するごとに を使用する必要はありません。また、Designer を使用して、JMS トリガーがメッセージを抽出できる宛先を作成する必要もありません。Integration

Server を使用してオンデマンドで管理オブジェクトを作成する方法は便利であり、開発サイクルでの時間節約にもつながります。

たとえば、JMS メッセージをパブリッシュする JMS トリガーとサービスを含む Integration Server パッケージを受信すると仮定します。また、Integration Server にはパッケージのサービスとトリガーで使用される JMS 接続エイリアスと同じ名前を持つ JMS 接続エイリアスが含まれ、JMS 接続エイリアスはオンデマンドで管理オブジェクトを作成するよう設定されており、Integration Server に実行中の Universal Messaging サーバを指し示す JNDI エイリアスがあると仮定します。JMS 接続エイリアスを開始すると、Integration Server では接続ファクトリが見つからない場合、自動的に作成されます。パッケージの送信サービスと JMS トリガーを使用すると、Integration Server ではサービスとトリガーに必要な宛先が見つからない場合、自動的に作成されます。

Integration Server を使用してオンデマンドで管理オブジェクトを作成する場合、以下の点に留意してください。

- オンデマンドで管理オブジェクトを作成するには、JNDI プロバイダが Universal Messaging JNDI プロバイダであり、JMS プロバイダが Universal Messaging であることが必要です。これは、Integration Server では JNDI プロバイダエイリアスの URL を使用して JNDI ネームスペースと Universal Messaging にオブジェクトを作成するためです。
- Universal Messaging JNDI プロバイダを含む多くの JNDI プロバイダでは、接続ファクトリ名と宛先名の大文字と小文字を区別します。つまり、myFactory と MYFactory は異なる 2 つのオブジェクトになります。
- Integration Server には、オブジェクトを作成する機能のみがあります。Integration Server は追加したオブジェクトの更新または削除を行うことはできません。
- Integration Server は、トランザクションタイプ XA_TRANSACTION の JMS 接続エイリアスで指定された接続ファクトリを作成する場合、XAConnectionFactory (javax.jms.XAConnectionFactory) を作成します。

  注:
  トランザクションタイプ LOCAL_TRANSACTION の JMS 接続エイリアスの場合、Integration Server は通常の接続ファクトリ (javax.jms.ConnectionFactory) を作成します。
- 接続ファクトリは存在するが、JMS 接続エイリアスで使用されるトランザクションタイプに一致しない場合、Integration Server は新しい接続ファクトリを作成しません。たとえば、JMS 接続エイリアスがトランザクションタイプ XA_Transaction および接続ファクトリ検索名 myConnectionFactory を指定するとします。さらに、myConnectionFactory という名前の javax.jms.ConnectionFactory オブジェクトが JNDI ネームスペースに既に存在するとします。Integration Server は JMS 接続エイリアスを開始し、JNDI ネームスペースで接続ファクトリを検索すると、既存の myConnectionFactory javax.jms.ConnectionFactory オブジェクトを検出します。Integration Server ではオブジェクトを削除したり、再作成して javax.jms.XAConnectionFactory にすることはありません。
- Integration Server では、Universal Messaging によって提供される水平スケーラビリティ機能を使用する接続ファクトリを自動的に作成しません。水平スケーラビリティ機能で使用可能な機能を利用する接続ファクトリを JMS 接続エイリアスで使用する場合、Universal Messaging で接続ファクトリを作成する必要があります。

- Integration Server が管理オブジェクトを検索した時に、そのオブジェクトが存在しない場合、Integration Server により管理オブジェクトが作成されます。管理オブジェクトの名前が正しく指定されていない場合、オンデマンドで管理オブジェクトを追加すると、不要な管理オブジェクトが作成されることがあります。たとえば、JMS 接続エイリアスに対して [**オンデマンドで管理オブジェクトを作成**] チェックボックスをオンにしており、myQueue を宛先として指定する pub.jms:send サービス、my_queue を指定する pub.jms:createConsumer サービスのインスタンス、および MyQueue を宛先として指定する JMS トリガーでこのエイリアスが使用されている場合、Integration Server は JNDI ネームスペースに 3 つの一意の宛先を作成します。

注:
Software AG では、実稼働環境でこの機能を使用しないことをお勧めします。つまり、実稼働環境で実行されている Integration Server の JMS 接続エイリアスに対して、[**オンデマンドで管理オブジェクトを作成**] (**Universal Messaging**) チェックボックスをオフのままにします。

接続ファクトリオブジェクトの変更の監視

接続ファクトリオブジェクトを使用して webMethods Broker への接続を確立する JMS 接続エイリアスの作成および編集時に、Integration Server で接続ファクトリの変更を監視するかどうかを指定できます。

Integration Server で接続ファクトリオブジェクトを使用して webMethods Broker への接続を確立する場合、Integration Server は JNDI 検索を使用して接続を作成します。通常、接続の確立後、Integration Server は接続が再開されるまで接続ファクトリオブジェクトを再検索しません。このため、接続の再開と再開の間は、クラスタポリシーの変更やクラスタ内の webMethods Broker の変更など、接続ファクトリオブジェクトに対して実行された可能性のある変更が接続に反映されません。接続ファクトリを監視することによって、接続ファクトリの変更後、関連する接続を自動的に Integration Server で更新できるようになります。

以下のいずれかの方法で、接続ファクトリオブジェクトを監視するように Integration Server を設定できます。

- **変更ポーリング**。Integration Server は指定されている JNDI プロバイダの接続ファクトリオブジェクトを定期的に検索して、接続の確立に使用した接続ファクトリオブジェクトと比較します。変更が検出されると、Integration Server は自動的に接続をリフレッシュします。
- **変更受信待機**。Integration Server は JNDI プロバイダにイベントリスナーを登録し、クラスタ接続ファクトリへの変更に関する通知を受信します。変更の通知を受信すると、Integration Server は自動的に接続をリフレッシュします。

変更を監視する方法に関係なく、Integration Server は、JMS メッセージを送信するサービスまたは JMS メッセージを受信するトリガーを中断せずに、接続の更新およびリフレッシュを試行します。ここでは、監視オプションの詳細について説明します。

注:
webMethods Broker は廃止されました。

変更ポーリング

JNDI プロバイダに対して変更をポーリングすることで、アクティブな接続に使用されている接続ファクトリを監視するように、Integration Server を設定できます。上記の場合、Integration Server は定期的に、接続の作成に使用されている接続ファクトリと JNDI プロバイダの接続ファクトリとを比較します。変更が検出されると、Integration Server は自動的に新しい接続ファクトリ定義を使用して接続をリフレッシュします。

Integration Server が JNDI プロバイダに対して接続ファクトリオブジェクトの変更をポーリングする頻度は、以下の 2 つの要因によって決まります。

- JMS 接続エイリアスに指定したポーリング間隔。JMS 接続エイリアスの作成または編集時にポーリング間隔を設定します。指定できる最小のポーリング間隔は 1 分です。また、webMethods 接続ファクトリオブジェクトに指定されているリフレッシュ間隔を使用するように Integration Server に指示することもできます。
- 接続監視期間。Integration Server がアクティブな webMethods Broker 接続の状態をチェックする頻度を決定します。接続監視期間は、watt.server.jms.connection.monitorPeriod 設定プロパティによって決まります。

接続監視期間が経過すると、Integration Server は JMS 接続エイリアスの状態をチェックします。また、Integration Server はポーリング間隔と最後のポーリングからの経過時間とを比較して、クラスタ接続ファクトリの変更をポーリングするかどうかを決定します。

Integration Server が、指定した間隔で接続ファクトリの変更をポーリングするように、ポーリング間隔の値は watt.server.jms.connection.monitorPeriod プロパティの値以上にする必要があります。ポーリング間隔の値が watt.server.jms.connection.monitorPeriod プロパティの値よりも小さかった場合、Integration Server は指定した間隔では接続ファクトリの変更をチェックしません。

たとえば、JMS 接続エイリアスでポーリング間隔を 1 分に指定し、watt.server.jms.connection.monitorPeriod は 10 分に設定したとします。ポーリング間隔のチェックおよび (必要に応じて) 変更ポーリングは、Integration Server が接続をチェックするときのみ発生するため、Integration Server はクラスタ接続ファクトリの変更を 10 分ごとにポーリングします。

注:
Integration Server と webMethods Broker 間の接続に失敗すると、Integration Server は watt.server.jms.connection.retryPeriod プロパティで決定された間隔で webMethods Broker への再接続を試行します。Integration Server によって接続が復元されると、Integration Server は直ちに JMS 接続エイリアスで指定されているクラスタ接続ファクトリの変更をポーリングします。

イベントリスナーの登録

JNDI プロバイダにイベントリスナーを登録することによって、JMS 接続エイリアスで使用されている接続ファクトリの変更を監視できます。具体的には、Integration Server は、JNDI ネームスペース内にある接続ファクトリオブジェクトの変更の通知を受けるイベントリスナーを登録します。イベントリスナーを登録すると、クラスタ接続ファクトリオブジェクトの変更イベントまたはエラーイベントに関する通知を Integration Server で受信できるようになります。

注:
Integration Server では、EventContext インタフェースをサポートしている JNDI プロバイダにイベントリスナーを登録できます。webMethods JNDI プロバイダは EventContext インタフェースをサポートしています。

変更イベントは、JMS 接続エイリアスで使用されている接続ファクトリオブジェクトで変更が発生したことを示します。Integration Server で接続ファクトリオブジェクトの変更イベントが受信されると、Integration Server は新しいクラスタ接続ファクトリオブジェクトを使用して webMethods Broker への接続をリフレッシュします。

エラーイベントは、JNDI プロバイダで NamingExceptionEvent が発生したことを示します。NamingExceptionEvent は複数の理由で発生する可能性があります。たとえば、Integration Server と JNDI プロバイダ間の接続に失敗した場合などに発生します。NamingExceptionEvent が発生すると、JNDI プロバイダはイベントリスナーの登録を取り消します。webMethods Broker 接続の状態の監視の一環として、Integration Server はイベントリスナーの状態を監視します。Integration Server によってイベントリスナーの登録取り消しが確認されると、Integration Server は変更リスナーを再登録して接続ファクトリオブジェクトの変更をポーリングし、必要に応じて、接続をリフレッシュします。

注:
watt.server.jms.connection.monitorPeriod 設定プロパティの値によって、Integration Server がアクティブな接続および登録されている変更リスナーの状態をチェックする頻度が決まります。デフォルト値は 45 秒です。

Integration Server による接続の更新方法

Integration Server によってクラスタ接続ファクトリの変更が確認されると、Integration Server は、関連する JMS 接続エイリアスに確立されている接続と関連するすべてのアクティビティを一時停止します。その後、Integration Server は接続を更新します。処理中のメッセージが失われないように、また重複メッセージが発生しないように、Integration Server は接続の更新の一環として以下の一連のタスクを完了します。

1. 再開が必要な接続を使用しているすべての JMS トリガーを一時停止します。JMS トリガーはこれ以上 webMethods Broker からメッセージを抽出しなくなりますが、既に抽出済みのメッセージの処理は続行します。JMS トリガーによって抽出済みのすべてのメッセージが処理されると、Integration Server は JMS トリガーを無効にします。

2. 再開が必要な接続の JMS 接続エイリアスを使用して JMS メッセージを送信しているサービスの新規スレッドを、一時的にブロックします。具体的には pub.jms:send、pub.jms:sendAndWait および pub.jms:reply です。

   watt.server.jms.connection.update.blockingTime サーバ設定パラメータの値は、pub.jms* サービスで使用している接続が更新されている間、サービスが接続を待機する最大時間を指定します。デフォルトは 1000 ミリ秒です。ブロック時間が経過するまでに Integration Server によって接続が再開されなかった場合、Integration Server は ISRuntimeException をスローします。

3. 接続を使用して JMS メッセージを送信している最中の pub.jms* サービスが実行を完了するまで待機します。

watt.server.jms.connection.update.restartDelay サーバ設定パラメータの値によって、接続を再開する前に、JMS メッセージを送信しているサービスが実行を完了するまで Integration Server が待機する時間が決まります。pub.jms* サービスが実行を完了する前に再開の遅延時間が経過した場合、Integration Server は ISRuntimeException をスローします。

4. JMS 接続エイリアスの接続を停止します。
5. 新しいクラスタ接続ファクトリオブジェクトを使用して接続をリフレッシュします。接続が有効にリフレッシュされると、JMS トリガーが有効になります。
6. pub.jms* サービスのスレッドのブロックを解除します。

接続ファクトリオブジェクトを監視するための Integration Server の設定

JMS 接続エイリアスで使用されている接続ファクトリオブジェクトを監視するように Integration Server を設定するときは、以下の点に留意してください。

- JNDI サーバ内の webMethods 接続ファクトリオブジェクトのみ監視できます。
- コンポジットクラスタ接続ファクトリオブジェクトを監視する場合、Integration Server はコンポジットクラスタ接続ファクトリオブジェクトの変更のみを監視します。Integration Server は、構成するクラスタ接続ファクトリオブジェクトの変更は監視しません。
- 変更リスナーを使用して接続ファクトリを監視するには、JNDI プロバイダで EventContext インタフェースをサポートしている必要があります。
- クラスタまたはコンポジット接続ファクトリのみを監視する場合は、[**変更ポーリング (ポーリング間隔は webMethods 接続ファクトリにより定義される)**] オプションを使用するように Integration Server を設定できます。
- 変更をポーリングすることで接続ファクトリを監視する場合、Integration Server が変更をポーリングする頻度は接続監視期間によって決まります。Integration Server は接続監視期間で決定された頻度で接続をチェックします。接続チェックの一環として、Integration Server はポーリング間隔が経過しているかどうかを確認します。経過している場合、Integration Server はクラスタ接続ファクトリの変更をポーリングします。ポーリング間隔が (watt.server.jms.connection.monitorPeriod パラメータで制御される) 接続監視期間よりも短い場合、Integration Server はポーリング間隔で指定されている頻度では変更をポーリングしません。
- JMS 接続エイリアスは、編集する前に無効にする必要があります。

> **注:**
> 以下の手順では、既に存在している JMS 接続エイリアスに監視を設定する方法を示します。エイリアスの作成時に監視を設定することもできます。

接続ファクトリを監視するように Integration Server を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**メッセージング**] に移動します。

3. [JMS 設定] の下の [**JMS 設定**] をクリックします。

4. [**JMS 接続エイリアスの定義**] で、関連する接続ファクトリの変更を監視する JMS 接続エイリアスを選択します。

5. [**JMS 接続エイリアスの編集**] をクリックします。

6. [接続プロトコル設定] の [**webMethods 接続ファクトリの監視**] リストから、以下のいずれかを選択します。

選択項目	目的
[いいえ]	Integration Server で接続ファクトリを監視しないことを指定します。これがデフォルトです。
[変更ポーリング (ポーリング間隔を指定する)]	指定した間隔で変更をポーリングすることで接続ファクトリを監視します。
[変更ポーリング (ポーリング間隔は webMethods 接続ファクトリにより定義される)]	webMethods 接続ファクトリオブジェクトに指定したリフレッシュ間隔によって決定される間隔で接続ファクトリを監視します。クラスタ接続ファクトリの設定の詳細については、『*Administering webMethods Broker*』および『*webMethods Broker Messaging Programmer's Guide*』を参照してください。
[変更リスナーの登録]	イベントリスナーを登録することで接続ファクトリを監視します。

7. [**変更ポーリング (ポーリング間隔を指定する)**] を選択した場合、[**ポーリング間隔 (分)**] でポーリング試行の間隔 (分単位) を指定します。ポーリング間隔は正の整数にする必要があります。デフォルト値は 60 分です。

8. [**変更内容の保存**] をクリックします。

メモ

Integration Server によって JMS 接続エイリアスが開始されると、Integration Server は以下の処理を実行します。

- JMS 接続エイリアスが webMethods Broker に接続しているかどうかを確認します。
- JMS 接続エイリアスによってイベントリスナー経由で接続ファクトリが監視されている場合、JNDI プロバイダで EventContext インタフェースがサポートされていることを確認します。

上記の条件のいずれかが満たされていない場合、Integration Server は JMSSubsystemException をスローします。

JMS での SSL の使用

Integration Server と JMS プロバイダ間の接続を SSL で保護する場合は、JMS プロバイダで SSL を設定して有効にする必要があります。

JNDI を使用して JMS プロバイダに接続する場合 (JMS 接続エイリアスで JNDI 検索を指定している場合) は、JMS プロバイダで SSL を設定する必要があります。具体的には、Integration Server を JMS プロバイダに接続するために使用する接続ファクトリを設定します。詳細については、JMS プロバイダのドキュメントを参照してください。

一部の JMS プロバイダでは、JMS クライアントが SSL ハンドシェーク に JVM のデフォルトの SSL コンテキストを使用する必要があります。この場合、JMS クライアントは、JVM 用の javax.net.ssl プロパティを使用して、キーストアの場所、トラストストアの場所、およびパスワード情報を設定する必要があります。ただし、これらのプロパティはストリング値をとり、結果的にファイルシステムのどこかにプレーンテキストでパスワード情報が保存されてしまいます。パスワード情報をプレーンテキストで保存したくない場合は、デフォルトの SSL コンテキストを確立するときに、JVM によって使用されるキーストアエイリアスまたはトラストストアエイリアス、もしくはその両方を割り当てることができます。起動時に、Integration Server は、キーストアエイリアスとトラストストアエイリアスからストアの場所とパスワードを取得することによって javax.net.ssl プロパティを設定した後、デフォルトの SSL コンテキストを作成します。詳細については、498 ページの「Integration Server JVM の SSL 情報の設定」を参照してください。

ネイティブ webMethods API を使用して webMethods Broker に接続する場合は、JMS 接続エイリアスの作成時に SSL 情報 (キーストア、キーストアタイプ、トラストストアおよびトラストストアタイプ) を設定します。

注:
Integration Server が JNDI を使用して webMethods Broker に接続する場合は、webMethods Broker で接続ファクトリを設定する必要があります。詳細については、*Administering webMethods Broker*を参照してください。

サポートされている JMS プロバイダ

Integration Server は次の JMS プロバイダでの使用が認定されています。

JMS プロバイダ	バージョン
メッセージングプロバイダ RabbitMQ および Azure Service Bus を使用する AMQP	AMQP 0-8、0-9、0-9-1、0-10
Apache ActiveMQ Classic	5.16.*
IBM MQ	9.0
JBoss Messaging	1.4.0 SP3 以上、2.0 未満
Oracle Streams Advanced Queuing (AQ)	10.2.*、11.2.0、12.2.0

JMS プロバイダ	バージョン
Software AG Universal Messaging	10.7 以上
Solace PubSub+	10.6.0
SonicMQ	7.5、7.6
WebLogic	9.1、9.2、10.3、12.1.3
webMethods Broker	6.5 以上 webMethods Broker は廃止されました。
WebSphere Application Server	8.5

JMS プロバイダに関する考慮事項

JMS サポートの特定の考慮事項については、JMS プロバイダのマニュアルを確認してください。

AMQP メッセージングプロバイダ

Integration Server は、RabbitMQ および Azure Service Bus プロバイダとの通信が認定されています。Integration Server は AMQP 用の QPID JMS クライアントライブラリを使用して、これらの JMS プロバイダと通信します。Integration Server は Qpid JMS AMQP 0-x 6.3.3 JMS クライアントライブラリでテストされました。

Apache ActiveMQ

SSL/TLS を使用して Apache ActiveMQ ブローカーに接続する場合、次の作業を行ってください。

- Apache ActiveMQ で提供されるキーツールを使用して、ブローカー用の認証を作成し、キーとトラストストアを設定し、ActiveMQ ブローカーの公開認証を Integration Server にインポートします。
- `javax.net.ssl` プロパティを使用して、ActiveMQ ブローカーとの接続用のキーストアファイルおよびトラストストアファイルの場所を設定します。Integration Server の場合、`custom_wrapper.conf` ファイルの `javax.net.ssl` プロパティを設定します。次に例を示します。
  - wrapper.java.additional.*n*= -Djavax.net.ssl.keystore=*keystoreLocation*
  - wrapper.java.additional.*n+1*= -Djavax.net.ssl.keyStorePassword=*password*
  - wrapper.java.additional.*n+2*= -Djavax.net.ssl.trustStore=*truststoreLocation*

  Microservices Runtime の場合、 *Integration Server_directory* `/bin/setenv.bat(sh)` ファイルで `JAVA_CUSTOM_OPTS` プロパティを使用して Java システムプロパティを設定します。
- ActiveMQ ブローカーを使用して双方向 SSL を設定する必要がある場合、Integration Server の公開認証を ActiveMQ ブローカートラストストアにインポートします。

IBM MQ

Integration Server は、IBM MQ 9.0 JMS プロバイダとの通信が認定されています。IBM MQ 9.0 に接続するときは、以下の点を考慮してください。

- IBM MQ 9.0 への SSL 接続を設定する場合、Integration Server に指定された JRE および IBM の推奨に基づいて、JNDI `.bindings` ファイルの CipherSuite 値を変更することを検討してください。
- デフォルトで、IBM MQ 9.0 は JMS 2.0 仕様をサポートします。ただし、Integration Server は JMS 1.1 仕様をサポートします。したがって、Integration Server が IBM MQ 9.0 とシームレスに動作するよう、 *Software AG_directory* `\profiles\IS_`*instance_name*`\configuration\custom_wrapper.conf` ファイルで次のプロパティを「`true`」に設定してください。

  wrapper.java.additional.*availableNumber*=-Dcom.ibm.mq.jms.SupportMQExtensions=true

  または、`watt.config.systemProperties` プロパティの値を `com.ibm.mq.jms.SupportMQExtensions=true` として設定できます。

  Microservices Runtime の場合、 *Integration Server_directory* `/bin/setenv.bat(sh)` ファイルで `JAVA_CUSTOM_OPTS` プロパティを使用して Java システムプロパティを設定します。

Solace PubSub+ Appliance および Cloud

Solace PubSub+ の直接転送モードでは次の機能をサポートしていません。

- メッセージセレクタ
- ローカルトランザクションのサポート
- 継続的サブスクリプション

注:
Solace PubSub+ の保証付き転送モードでは上記の機能をサポートしています。

Solace PubSub+ の直接転送モードを使用し、JMS プロバイダが上記のいずれかの機能を使用する場合、Integration Server は次の方法で応答します。

- JMS トリガーがトピックにサブスクライブされ、メッセージセレクタ機能を使用する場合、Integration Server はトリガーを有効にせず、「Error creating consumer - cannot use selector with direct messaging」というメッセージを表示します。
- Solace PubSub+ をポイントする JMS 接続エイリアスのトランザクションタイプがローカルトランザクション または XA トランザクションである場合、Integration Server は JMS の送受信操作を実行できず、「Transacted sessions or XA sessions are not supported with direct transport」というメッセージを表示します。
- JMS トリガーが継続的サブスクリプション名を持つトピックにサブスクライブされた場合、Integration Server はトリガーを有効にせず、「Unknown Durable Topic Endpoint」というメッセージを表示します。

JMS プロバイダとしての Software AG Universal Messaging の使用について

JMS プロバイダとして Universal Messaging を使用する場合は、以下の点に留意してください。

- Universal Messaging を使用するときに、 Universal Messaging のバージョンが、Integration Server のバージョン以上である場合、Integration Server のクラスパスにクライアントライブラリを追加する必要はありません。
- Universal Messaging クライアントライブラリを最新の状態にしておきます。クライアントライブラリを更新する Universal Messaging の修正をインストールする場合は、更新された Universal Messaging クライアントライブラリファイルを、Integration Server が使用する *Software AG_directory* /common/lib ディレクトリにコピーして下さい。
- Integration Server をより新しいバージョンの Universal Messaging とともに使用する場合、Integration Server は、より新しいバージョンの Universal Messaging で提供される機能を使用できません。Integration Server は、同等のバージョンの Universal Messaging に存在する機能だけを使用できます。
- Integration Server では、永続メッセージを Universal Messaging にパブリッシュするときに、同期パブリッシュがデフォルトで使用されます。

  JMS プロバイダとして Universal Messaging を使用する場合、JMS クライアントは同期または非同期のパブリッシュを使用できます。同期パブリッシュでは、JMS クライアントはメッセージを Universal Messaging に 1 つずつ送信します。JMS クライアントは、メッセージが Universal Messaging にデリバーされた後にのみ、送信成功を示す応答を受信します。非同期パブリッシュでは、JMS メッセージは、バッファに配置された後、バッチで Universal Messaging に送信されます。ただし、JMS クライアントは、メッセージが実際に Universal Messaging にデリバーされる前に、送信成功を示す応答を受信することがあります。非同期パブリッシュは同期パブリッシュよりも高速ですが、Universal Messaging への接続が失敗するか、または Integration Server が使用できなくなると、メッセージが失われる可能性があります。JMS プロバイダとして Universal Messaging を使用する JMS クライアントのデフォルトは、非同期パブリッシュです。このデフォルトは、永続デリバーモードか非永続デリバーモードかに関係なく、すべてのメッセージに適用されます。ただし、永続メッセージのデリバーを確実にするために、Integration Server では常に同期パブリッシュを使用して永続 JMS メッセージを Universal Messaging に送信します。
- Universal Messaging が JMS プロバイダである場合、メッセージの優先順位はサポートされません。Universal Messaging に送信される JMS メッセージに割り当てられた優先順位は無視されます。
- Universal Messaging を JMS プロバイダとして使用し、継続的サブスクライバを使用する場合、Universal Messaging Enterprise Manager を使用して ConnectionFactory または TopicConnectionFactory を作成するときに、必ず [**Shared Durable**] オプションを選択してください。たとえば、ConnectionFactory を作成する場合、[**Connection Factory (Shared Durable)**] オプションを選択します。
- Integration Server の JMS 接続エイリアスが Universal Messaging 領域サーバのクラスタに接続されており、クラスタ内のマスター領域サーバが停止した場合、Integration Server はエイリアスを停止します。これにより、エイリアスに関連付けられたすべての JMS トリガー

は停止し、すべての JMS 送信サービスは ResourceUnavailableException で失敗します。クラスタ自体で問題を解決できる (つまり、クラスタで新しいマスター領域サーバを再選択できる) 場合、エイリアスはクラスタに自動的に再接続します。

- Integration Server が、PauseServerPublishing 設定パラメータが「true」に設定されている Universal Messaging サーバにドキュメントをパブリッシュすると、パブリッシュは失敗し nPublishPauseException が発生します。Integration Server では、メッセージング接続エイリアスを使用できないかのように処理を続行して nPublishPauseException を処理します。
- Universal Messaging が JMS プロバイダである場合、Integration Server は、JMS トリガーのポーリング間隔を制御するサーバ設定プロパティに最小値を強制的に適用します。Universal Messaging に接続する場合、これらのパラメータのデフォルト値は最適値ではありません。このデフォルト値を含めて、ポーリング間隔を小さい値にすると、Universal Messaging サーバで CPU 使用率が高くなり、Universal Messaging の使用時に、パフォーマンスが改善されない場合があります。

  Universal Messaging が JMS プロバイダである場合、ポーリング間隔を制御するサーバ設定パラメータの最小値は、以下の表のとおりです。

パラメータ	Universal Messaging で使用する最小値
watt.server.jms.trigger.concurrent.primaryThread.pollingInterval	3000 ミリ秒
watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval	同時実行 JMS トリガーが結合を使用しない場合は、3000 ミリ秒 同時実行 JMS トリガーが結合を使用する場合は、10 ミリ秒
watt.server.jms.trigger.serial.primaryThread.pollingInterval	3000 ミリ秒 **注:** 逐次実行 JMS トリガーが結合を使用する場合、Integration Server は 10 ミリ秒に設定されたセカンダリポーリング間隔を使用します。逐次実行 JMS トリガーに対して、セカンダリポーリング間隔を設定することはできません。

  サーバ設定パラメータを変更して、ポーリング間隔を大きい値に変更できます。指定した値がすべての JMS プロバイダで使用されることに留意してください。Universal Messaging に接続するときに、Integration Server は最小値を強制的に適用します。
- 結合を使用する JMS トリガーは、上記の最小値の場合でも、Universal Messaging の CPU 使用率に影響を及ぼす可能性があります。これは、セカンダリポーリング間隔 (10 ミリ秒) が低い値になっているためです。JMS トリガーが異なる宛先からメッセージを迅速に抽出できるように、セカンダリポーリング間隔は低くする必要があります。Universal Messaging に対する影響を小さくするには、以下のいずれかの手順に従います。
  - 同時実行 JMS トリガーの場合、トリガーで使用される JMS 接続エイリアスが複数の接続を使用するように設定して、次に JMS トリガーの [**接続数**] プロパティを宛先の数で等

しく分割できる値に設定します。たとえば、JMS トリガーが 2 つの宛先からメッセージを抽出する場合、[**接続数**] を 2、4、6、8... に設定します。JMS トリガーが 3 つの宛先からメッセージを抽出する場合、[**接続数**] を 3、6、9... に設定します。

- 逐次実行または同時実行 JMS トリガーの場合、JMS トリガー用の結合を使用するのではなく、代わりに Universal Messaging チャネル結合を使用して、異なる宛先から Universal Messaging サーバ上の 1 つの宛先にメッセージを収集します。

- Integration Server などの Universal Messaging クライアントは宛先オブジェクトに関する情報をキャッシュします。Integration Server はクライアントストアキャッシュにこの情報を保持します。ただし、Universal Messaging Enterprise Manager を使用して特定のキューまたはチャネルのプロパティ (廃棄までの時間、イベントの最大数など) を変更すると、Universal Messaging サーバはチャネルまたはキュー用の新しい宛先オブジェクトを作成します。ただし、Integration Server はチャネルまたはキューの新しい宛先オブジェクトを認識せず、キャッシュされた無効な宛先オブジェクトの使用を継続します。したがって、Integration Server は宛先を検出できないため、チャネルまたはキューへのメッセージの送信は失敗します。同じ理由で、JMS トリガーはチャネルまたはキューからメッセージを受信できません。この問題を解決し、Integration Server が宛先に接続できるようにするには、最初に JMS 接続エイリアスを無効にした後有効にすることで、エイリアスを再び開始する必要があります。

Integration Server クラスパスへの JMS プロバイダクライアントライブラリの追加

Integration Server から、JMS プロバイダの一部の Java クライアントライブラリにアクセスできる必要があります。以下のいずれかの方法を実行できます。

- 以下の手順で説明するように、 *Integration Server_directory* \instances*instance_name*\lib\jars\custom ディレクトリにライブラリを配置して、サーバのクラスパスに配置します。

- ライブラリを *Integration Server_directory* \lib\jars\custom ディレクトリ内に配置することで、これらのライブラリをすべてのサーバインスタンスが利用できるようにすることができます。

  注:
  サーバのクラスパスにあるファイルは、 *Integration Server_directory* \lib\jars\custom ディレクトリにあるファイルよりも優先します。

- *packageName*\code\jars ディレクトリに jar ファイルを配置して、パッケージクロスローダ内に隔離します。

  パッケージクラスローダ内にファイルを配置する場合は、この JMS プロバイダへの JMS 接続エイリアスを設定するときに、必ず [**クラスローダ**] プロパティを設定するようにしてください。

JMS プロバイダの .jar ファイルは、Integration Server で動作する他の webMethods コンポーネントと競合する可能性があります。競合が発生した場合は、代わりにパッケージクラスローダを使用することをお勧めします。

注:

webMethods Broker および JNDI プロバイダの .jar ファイルは既に Integration Server に含まれています。Universal Messaging .jar ファイルは *Software AG_directory* /common/lib に含まれています。

サードパーティの JMS プロバイダの JMS および JNDI ライブラリを Integration Server クラスパスに追加するには

1. JMS プロバイダの Java API ライブラリを、以下に指定されている場所から *Integration Server_directory* \instances*instance_name*\lib\jars\custom ディレクトリにコピーします。

JMS プロバイダ	コピーするファイル
AMQP メッセージ形式	Apache QPID ライブラリから: apache-qpid-jms-amqp-0-x-6.3.3-bin.tar.gz または apache-qpid-jms-amqp-0-x-6.3.3-bin.zip
Apache ActiveMQ Classic	■ *ActiveMQ_HOME*\activemq-all-5.16.*.jar *ActiveMQ_HOME* は、Apache ActiveMQ がインストールされたディレクトリです。例: apache-activemq-5.16.1-bin\apache-activemq-5.16.1
IBM MQ	IBM MQ バージョン 9.0 の場合 ■ *IBMMQ*\Java\lib\com.ibm.mq.allclient.jar ■ *IBMMQ*\Java\lib\fscontext.jar ■ *IBMMQ*\Java\lib\providerutil.jar ■ *IBMMQ*\Java\lib\jms.jar *IBMMQ* は IBM MQ がインストールされているディレクトリです。
JBoss Messaging	■ jboss-messaging-client.jar (JBoss Messaging の配布ファイルに含まれています。) ■ *JBOSS_HOME*\client\jbossall-client.jar ■ *JBOSS_HOME*\server*SERVER_NAME*\deploy\jboss-aop.deployer\jboss-aop.jar ■ *JBOSS_HOME*\server*SERVER_NAME*\lib\javassist.jar ■ *JBOSS_HOME*\server*SERVER_NAME*\lib\trove.jar *JBOSS_HOME* は JBoss がインストールされているディレクトリで、*SERVER_NAME* はメッセージングサーバの名前です。 注:

JMS プロバイダ	コピーするファイル
	JBoss Messaging 1.4.0 には jboss-remoting.jar のパッチバージョンが必要です。
Oracle Streams Advanced Queuing (AQ)	Oracle Streams Advanced Queuing (AQ) 10.2.* の場合 ■ *ORACLE_HOME*\jdbc\lib\classes12.jar ■ *ORACLE_HOME*\jlib\orai18n.jar ■ *ORACLE_HOME*\lib\xmlparserv2.jar ■ *ORACLE_HOME*\rdbms\jlib\xdb.jar ■ *ORACLE_HOME*\rdbms\jlib\aqapi13.jar ■ *ORACLE_HOME*\rdbms\jlib\jmscommon.jar *ORACLE_HOME* は *ORACLE_BASE*\product\10.2.0\db_1 で、*ORACLE_BASE* は Oracle データベースがインストールされているディレクトリです。 Oracle Streams Advanced Queuing (AQ) 11.2.0 の場合 ■ *ORACLE_HOME*\jlib\orai18n.jar ■ *ORACLE_HOME*\lib\xmlparserv2.jar ■ *ORACLE_HOME*\rdbms\jlib\xdb.jar ■ *ORACLE_HOME*\rdbms\jlib\aqapi.jar ■ *ORACLE_HOME*\rdbms\jlib\jmscommon.jar ■ *ORACLE_HOME*\rdbms\jlib\ ojdbc6.jar *ORACLE_HOME* は *ORACLE_BASE*\product\11.2.0\db_1 で、*ORACLE_BASE* は Oracle データベースがインストールされているディレクトリです。 **注:** JMS プロバイダとして Oracle Streams Advanced Queuing (AQ) 11.2.0 を使用する場合は、watt.config.systemProperties プロパティの値に「`oracle.jms.conservativeNavigation=true`」が含まれている必要があります。 Oracle Streams Advanced Queuing (AQ) 12.2.0 の場合 ■ *ORACLE_HOME*\jdbc\lib\ojdbc8.jar ■ *ORACLE_HOME*\jlib\orai18n.jar ■ *ORACLE_HOME*\lib\ xmlparserv2.jar

JMS プロバイダ	コピーするファイル
	■ *ORACLE_HOME*\rdbms\jlib\aqapi.jar ■ *ORACLE_HOME*\rdbms\jlib\jmscommon.jar ■ *ORACLE_HOME*\rdbms\jlib\ xdb.jar *ORACLE_HOME* は *ORACLE_BASE*\product\12.2.0\db_1 で、*ORACLE_BASE* は Oracle データベースがインストールされているディレクトリです。
Solace PubSub+	Solace ライブラリからまたは Solace インストールディレクトリから: ■ sol-common-10.6.0.jar ■ sol-jcsmp-10.6.0.jar ■ sol-jms-10.6.0.jar
SonicMQ	SonicMQ バージョン 7.5 の場合 ■ *SonicMQ_directory*\lib\mfcontext.jar ■ *SonicMQ_directory*\lib\sonic_Client.jar ■ *SonicMQ_directory*\lib\sonic_Crypto.jar ■ *SonicMQ_directory*\lib\sonic_mgmt_client.jar ■ *SonicMQ_directory*\lib\sonic_Selector.jar ■ *SonicMQ_directory*\lib\sonic_XA.jar SonicMQ バージョン 7.6 の場合、上記に加えて下記も必要 ■ *SonicMQ_directory*\lib\mgmt_client.jar ■ *SonicMQ_directory*\lib\mgmt_config.jar ■ *SonicMQ_directory*\lib\sonic_XMessage.jar *SonicMQ_directory* は SonicMQ がインストールされているディレクトリです。
WebLogic	WebLogic 9.1 および 9.2 の場合 ■ *WebLogic_directory*\server\lib\weblogic.jar WebLogic 10.3 および 12.1.3 の場合 ■ *WebLogic_directory*\server\lib\wljmsclient.jar ■ *WebLogic_directory*\server\lib\wlnmclient.jar ■ *WebLogic_directory*\server\lib\wlclient.jar

JMS プロバイダ	コピーするファイル
	WebLogic_directory は WebLogic がインストールされているディレクトリです。
WebSphere Application Server	WebSphere Application Server バージョン 8.5 の場合 ■ *WebSphereApplicationServer*\AppServer\runtimes\com.ibm.ws.ejb.thinclient_8.5.0.jar ■ *WebSphereApplicationServer*\AppServer\runtimes\com.ibm.ws.orb_8.5.0.jar ■ *WebSphereApplicationServer*\AppServer\runtimes\com.ibm.ws.sib.client.thin.jms_8.5.0.jar *WebSphereApplicationServer* は WebSphere Application Server がインストールされているディレクトリです。

2. Integration Server を再起動します。

注:
各 JMS プロバイダのファイルリストは一般的なガイドラインです。各 JMS プロバイダの要件は変更されることがあります。必ず JMS プロバイダのドキュメントを参照して、必要な jar ファイルの正確なリストを確認するようにしてください。さらに、選択した JMS プロバイダのマニュアルを確認し、JMS プロバイダの Java クライアントライブラリへの修正と更新の処理方法を決定します。Java クライアントライブラリへの更新または修正は、Integration Server クラスパスに追加されたプロバイダクライアントライブラリへの更新を要求する場合があります。

15 MQTT メッセージングのための Integration Server の設定

- Integration Server の MQTT サポートの概要 348
- サポートされている MQTT サーバ 349
- MQTT 接続エイリアスの作成 350
- MQTT 接続エイリアスの編集 354
- MQTT 接続エイリアスの有効化および無効化 355
- MQTT 接続エイリアスの削除 356
- MQTT サーバへの失敗した接続の再試行間隔の指定 356

Integration Server の MQTT サポートの概要

MQTT は、IoT (Internet of Things: モノのインターネット) および M2M (machine-to-machine: マシン間) 接続において、デバイスと共に使用されることが多いパブリッシュ/サブスクライブメッセージングプロトコルです。MQTT は軽量であり、スマートデバイスのセンサーなど、リソースとネットワーク帯域幅が制約されたデバイスで使用するよう設計されました。このプロトコルは、センサーおよびデバイスからのデータを計測し送信するテレメトリーアプリケーションをサポートします。

注:
MQTT の機能の詳細な説明はこのマニュアルの範囲外ですが、別途入手できます。

他のパブリッシュ/サブスクライブソリューションと同様、MQTT はメッセージのパブリッシュまたはサブスクライブを行うクライアント、およびパブリッシュされたメッセージをサブスクライバにルーティングするサーバから構成されます。

MQTT バージョン 3.1.1 をサポートする Integration Server は、MQTT サーバでトピックにメッセージをパブリッシュする MQTT クライアント、および MQTT サーバでトピックにサブスクライブする MQTT クライアントとして機能することができます。具体的には以下のとおりです。

- Integration Server は組み込みサービス pub.mqtt:publish を使用して MQTT メッセージを MQTT サーバにパブリッシュできます。
- Integration Server は MQTT トリガーを作成することでトピックにサブスクライブできます。MQTT トリガーは MQTT サーバのトピックにパブリッシュされたメッセージを受信し、トリガーサービスを呼び出してメッセージを処理します。
- Integration Server は MQTT 接続エイリアスを使用して、MQTT サーバへの接続を作成します。pub.mqtt:publish サービスを呼び出すには、MQTT 接続エイリアスが MQTT サーバにメッセージをパブリッシュする必要があります。同様に、MQTT トリガーは MQTT 接続エイリアスを指定します。MQTT トリガーはこのエイリアスを使用して、メッセージの受信元でありトリガーがサブスクリプションを作成する先である MQTT サーバを指定します。

注:
また、MQTT サーバは MQTT ブローカと呼ばれることもあります。Integration Server のマニュアルでは、webMethods Broker 製品との混同を避けるため、MQTT サーバという用語を使用します。

Integration Server への MQTT ソリューションの追加は、次の表に示す基本タスクから構成されます。

ステージ 1	使用する MQTT プロバイダを決定します。サポートされている MQTT プロバイダの一覧については、349 ページの「サポートされている MQTT サーバ」を参照してください。
ステージ 2	1 つ以上の MQTT 接続エイリアスを作成します。MQTT 接続エイリアスの作成については、350 ページの「MQTT 接続エイリアスの作成」を参照してください。

ステージ 3	MQTT トリガーを作成します。Integration Server を使用して MQTT サーバから MQTT メッセージを抽出し処理する場合にのみ、このステップは必要です。 MQTT トリガーの作成の詳細については、*webMethods Service Development Help*を参照してください。
ステージ 4	pub.mqtt:publish サービスを使用して、MQTT メッセージをパブリッシュするサービスを作成します。Integration Server を使用して MQTT メッセージをパブリッシュする場合にのみ、このステップは必要です。pub.mqtt:publish サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

Integration Server の MQTT サポートの制限

Integration Server における MQTT サポートには次の制限があります。

- 拡張ロギングは、Integration Server がメッセージの送信、受信、または処理を行うときに詳細なログエントリを書き込むよう指示しますが、MQTT と共に使用することはできません。
- クライアントサイドキュー (CSQ) は、Integration Server の他のタイプのメッセージング接続エイリアスでは利用可能な機能ですが、MQTT 接続エイリアスにはありません。
- 重複抑制メッセージ処理機能は MQTT トリガーでは利用できません。
- Integration Server のクラスタの各ノードで、同名の MQTT トリガーと MQTT 接続エイリアスが存在することはできません。さらに、クラスタのすべての MQTT 接続エイリアスは一意の接続クライアント ID を使用する必要があります。重複する接続クライアント ID は使用しないでください。

  このようなクラスタの制限があるのは、MQTT 仕様で複数クライアントによるサブスクリプションの共有がサポートされていないためです。同一の MQTT トリガーは同じサブスクリプションと同じ接続クライアント ID を共有します。MQTT トリガーの接続クライアント ID は、MQTT 接続エイリアスの接続クライアント ID 値にトリガー名が追加されたものです。

  MQTT 仕様への準拠のため、一部の MQTT プロバイダは共有サブスクリプションおよび複数クライアントによる同一接続クライアント ID の使用をサポートしていません。この制限の回避策を提供する MQTT プロバイダもあります。

サポートされている MQTT サーバ

Integration Server は MQTT バージョン 3.1.1 をサポートしています。Integration Server は、次の表に示す MQTT サーバとの併用が認定されています。

MQTT サーバ	バージョン
Eclipse Mosquitto	1.5、1.6
Universal Messaging	10.5、10.7

Eclipse Paho ライブラリバージョン 1.2.1 には、MQTT メッセージングプロトコルのクライアント実装とユーティリティが含まれています。このライブラリは Integration Server で *Software AG_directory* /common/lib にインストールされています。Paho ライブラリは Universal Messaging と Mosquitto への接続に使用できます。

MQTT 接続エイリアスの作成

MQTT 接続エイリアスには、Integration Server が MQTT サーバへの接続を確立するために必要な設定情報が含まれています。Integration Server は MQTT メッセージのパブリッシュ時に MQTT 接続エイリアスを使用します。MQTT トリガーは MQTT 接続エイリアスを使用して、MQTT サーバのサブスクライブ先のトピックからメッセージを抽出します。各 Integration Server にはそれぞれ複数の MQTT 接続エイリアスを設定できます。

MQTT 接続エイリアスが MQTT サーバのセキュアポートへの接続を確立する場合、エイリアスの設定を完了する前に以下の手順に従う必要があります。

- MQTT サーバとの通信に使用する認証局 (CA) の信用のある認証を含むトラストストアを作成します。トラストストア用のトラストストアエイリアスを作成します。

  トラストストアは、一方向および双方向 (相互) の SSL 接続に必要です。

- MQTT サーバのポートが双方向 SSL 用に設定されている場合、以下の手順に従います。
  - Integration Server が MQTT サーバとの接続に提供するクライアント認証を含むキーストアを作成します。
  - キーストアのキーストアエイリアスを作成し、MQTT サーバに安全に接続するために使用する秘密鍵および関連する認証を含むキーエイリアスを指定します。
- MQTT 接続エイリアスをセキュアな Universal Messaging ポートに設定する場合、トラストストアと、場合によってはキーストアエイリアスとキーエイリアスを指定する必要があります。
  - Universal Messaging ポートに [**Enable Client Cert Validation**] オプションがオンになっていない場合、トラストストアエイリアスを指定する必要があります。
  - Universal Messaging ポートに [**Enable Client Cert Validation**] オプションがオンになっている場合、トラストストアエイリアス、キーストアエイリアス、およびキーエイリアスを指定する必要があります。

SSL による Integration Server との安全な接続の確立の詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。トラストストアエイリアスとキーストアエイリアスの作成の詳細については、509 ページの「Integration Server におけるキーストアとトラストストアの使用」を参照してください。

MQTT 接続エイリアスを作成するには、以下の手順に従います。

MQTT 接続エイリアスを作成するには

1. Integration Server Administrator を開きます。

2. [**メッセージング**] > [MQTT 設定] に移動します。
3. [**MQTT 接続エイリアスの作成**] をクリックします。
4. [**全般設定**] で、次の表のフィールドの値を指定します。

フィールド	説明
[接続エイリアス名]	MQTT 接続エイリアスの一意の名前。各 MQTT 接続エイリアスには、Integration Server が MQTT サーバへの接続を確立するために必要な設定情報が含まれています。
[説明]	MQTT 接続エイリアスの説明。
[パッケージ]	MQTT 接続エイリアスに関連付けるパッケージの名前。Integration Server はこのパッケージに接続エイリアス設定を保存します。MQTT 接続エイリアスをパッケージに割り当てると、Integration Server はパッケージに関連してエイリアスを有効/無効にします。パッケージを無効にすると、Integration Server は MQTT 接続エイリアスを自動的に無効にします。パッケージが他のサーバにパブリッシュされた場合、パッケージはその MQTT 接続エイリアスを保持します。サブスクライバがパッケージをインストールした場合、MQTT 接続エイリアスがサブスクライバの Integration Server に自動的に作成されます。MQTT 接続エイリアスに割り当てるパッケージがない場合、[**パッケージ**] の設定は WmRoot (サーバのコアパッケージ) のままにします。これによって、Integration Server の実行中はいつでも MQTT 接続エイリアスを利用できます。

5. [**接続プロトコル設定**] で、次の表のフィールドの値を指定します。

フィールド	説明
[ホスト]	MQTT サーバのプロトコル、ドメイン、およびポートが入った URL。 MQTT 接続エイリアスが SSL を使用して MQTT サーバに接続するよう設定する必要がある場合、URL は ssl:// で始めてください。
[接続クライアント ID]	この MQTT 接続エイリアスに関連付けられた接続のクライアント識別子。Integration Server は接続クライアント識別子を自動的に生成しますが、独自の識別子を指定できます。
[サブスクリプションにクリーンなセッションを使用する]	この MQTT 接続エイリアスを使用して MQTT トリガーに対して作成された接続が MQTT サーバで永続的なセッションを確立するかどうかを示します。次のいずれかの手順に従います。

フィールド	説明
	■ この MQTT 接続エイリアスを使用する MQTT トリガーによって確立された以前のサブスクリプションを削除するには、[**サブスクリプションにクリーンなセッションを使用する**] チェックボックスをオンにします。セッションが終了すると、MQTT サーバは MQTT トリガーによって作成されたサブスクリプションを削除します。 ■ セッション間でサブスクリプションを継続させるには、[**サブスクリプションにクリーンなセッションを使用する**] チェックボックスをオフにします。セッション中に MQTT トリガーによって作成された新しいサブスクリプションが、トリガーの既存のサブスクリプションに追加されます。さらに、サブスクリプションはトリガーの切断後も残存します。QoS (Quality of Service : サービス品質) が 1 または 2 である場合、トリガーの切断時に送信されたメッセージは保持されます。トリガーは、再接続すると、取得されなかったメッセージを抽出します。
[接続タイムアウト (秒)]	Integration Server が MQTT サーバへのネットワーク接続の確立まで待機する最大秒数。値 0 を指定すると、タイムアウトが無効になります。つまり、Integration Server はネットワーク接続の試行が失敗または成功するまで待機します。デフォルトは 30 秒です。
[キープアライブ (秒)]	この MQTT 接続エイリアスに対して作成された接続を使用した、MQTT サーバからのメッセージの送信または受信間で経過可能な最大秒数。MQTT のキープアライブ機能によって、MQTT クライアント (IS) と MQTT サーバ間の接続がオープンしていることが確認されます。値 0 を指定すると、キープアライブは無効になります。デフォルトは 60 秒です。

6. [**Last-Will 設定**] で、MQTT サーバからの予期しない切断が発生した場合に Integration Server が示す情報を入力します。

フィールド	説明
[Last-Will の有効化]	Integration Server が MQTT サーバから突然切断された場合、Integration Server の代わりに MQTT サーバが will メッセージを送信するかどうかを指定します。will メッセージを使用すると、他の MQTT クライアントは突然の切断を処理できます。Integration Server が MQTT サーバから突然切断されたときに MQTT サーバで特定のトピックに特定のメッセージを送信する場合、[**Last-Will の有効化**] をオンにします。次のいずれかの手順に従います。

フィールド	説明
	■ Integration Server が予期せずに切断されたときに will メッセージを送信するには、[**Last-Will の有効化**] チェックボックスをオンにします。 ■ Integration Server が予期せずに切断されたときに will メッセージを送信しない場合は、[**Last-Will の有効化**] チェックボックスをオフにします。
[Last-Will QoS]	will メッセージのサービスの品質。次のいずれかを選択します。 ■ **[最大で 1 回 (0)]** ■ **[少なくとも 1 回 (1)]** ■ **[1 回のみ (2)]**
[Last-Will 保持]	MQTT サーバが Last-Will メッセージを保持するかどうかを指定します。次のいずれかの手順に従います。 ■ Last-Will メッセージを保持するよう MQTT サーバに指示するには、[**Last-Will 保持**] チェックボックスをオンにします。 ■ MQTT サーバが Last-Will メッセージを保持しない場合は、[**Last-Will 保持**] チェックボックスをオフにします。
[Last-Will トピック]	Last-Will メッセージをパブリッシュする先のトピック。このトピックのすべてのサブスクライバは Last-Will メッセージを受信します。MQTT 接続エイリアスの作成時にこのトピックが存在している必要はありません。Integration Server が最初に MQTT 接続エイリアスを使用して接続を確立したときにトピックが存在しない場合、MQTT サーバはトピックを作成します。
[Last-Will メッセージペイロード]	Last-Will の内容。
[接続時メッセージペイロード]	MQTT サーバへの接続成功時に送信するメッセージ。これは接続メッセージのペイロードであり、[**Last-Will トピック**] と同じトピックに送信されます。
[切断時メッセージペイロード]	MQTT サーバからの切断時に送信するメッセージ。これは切断メッセージのペイロードであり、[**Last-Will トピック**] と同じトピックに送信されます。

7. [**セキュリティ設定**] で、MQTT サーバへの接続に基本認証が必要な場合、Integration Server が接続の確立に使用するユーザ名とパスワードを指定します。

フィールド	説明
[ユーザ名]	クライアント認証に指定する Integration Server のユーザ名。
[パスワード]	ユーザ名のパスワード。

8. [**セキュリティ設定**] で、MQTT サーバへの接続に SSL が必要な場合、[**SSL を使用**] に [**はい**] を選択し、以下のトラストストアとキーストアの両方またはいずれか一方の情報を指定します。

フィールド	説明
[トラストストアエイリアス]	MQTT サーバの認証局 (CA) の認証を含むトラストストアのエイリアス。[**SSL を使用**] が [**はい**] に設定されている場合、トラストストアエイリアスを選択する必要があります。
[キーストアエイリアス]	Integration Server で相互 (双方向) SSL ハンドシェイク間に MQTT サーバに送信されるクライアント認証を含むキーストアのエイリアス。キーストアエイリアスの選択が必要となるのは、MQTT サーバのポートが相互 (双方向) SSL を使用するように設定される場合だけです。
[キーエイリアス]	Integration Server で相互 (双方向) SSL ハンドシェイク間に MQTT サーバに送信される Integration Server の秘密鍵および関連する認証のエイリアス。このキーエイリアスは、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。[**キーストアエイリアス**] を指定した場合は、[**キーエイリアス**] を指定する必要があります。

9. [**変更内容の保存**] をクリックします。

10. MQTT 接続エイリアスを有効にします。

MQTT 接続エイリアスの編集

MQTT 接続エイリアスの編集が必要になることがあります。たとえば、サブスクライバがクリーンセッションを使用しないよう、エイリアスを変更する場合があります。または、Integration Server が突然切断したときに Integration Server の代わりに MQTT サーバがそれ以上 Last-Will メッセージを送信しないように、MQTT エイリアスを変更する場合があります。

MQTT 接続エイリアスを編集する場合は、以下の情報に留意してください。

- MQTT 接続エイリアスは、編集する前に無効にする必要があります。
- MQTT 接続エイリアス名は編集できません。
- [**接続クライアント ID**] を変更すると、MQTT サーバでエイリアスを使用する MQTT トリガーのサブスクリプションが新規になることがあります。古い接続クライアント ID を使用するサブスクリプションは MQTT サーバから自動的に削除されません。古いサブスクリプションはメッセージを含み、メッセージの受信を続行することがあります。

MQTT 接続エイリアスを編集するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [MQTT 設定] に移動します。
3. [MQTT 接続エイリアスの定義] リストから、編集する MQTT 接続エイリアスを選択します。
4. [**MQTT 接続エイリアスの編集**] をクリックします。

   [**MQTT 接続エイリアスの編集**] リンクが利用できない (下線が引かれていない) 場合、[MQTT 設定] ページに戻り、エイリアスを無効にします。
5. MQTT 接続エイリアスのプロパティを編集します。
6. [**変更内容の保存**] をクリックします。
7. MQTT 接続エイリアスを有効にします。

MQTT 接続エイリアスの有効化および無効化

MQTT 接続エイリアスが有効な場合、Integration Server はエイリアスを使用して、MQTT サーバにメッセージをパブリッシュし、エイリアスを使用する MQTT トリガーの代わりにトピックへのサブスクリプションを作成し、MQTT トリガーがサブスクライブするトピックからメッセージを抽出できます。MQTT 接続エイリアスを無効な場合、Integration Server によって、エイリアスを使用するすべての MQTT トリガーが停止されます。トリガーステータスは [有効] になります (接続エイリアスが実行されていない)。さらに、エイリアスが無効な場合、MQTT 接続エイリアスを使用して MQTT サーバにメッセージをパブリッシュするサービスはエラーで終了します。

MQTT 接続エイリアスを有効または無効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [MQTT 設定] に移動します。
3. [MQTT 接続エイリアスの定義] リストの [**有効**] 列で以下のいずれかを行います。
   - エイリアスが無効になっており、このエイリアスを有効にする場合は、[**いいえ**] をクリックします。
   - エイリアスが有効になっており、このエイリアスを無効にする場合は、[**はい**] をクリックします。

MQTT 接続エイリアスの削除

MQTT 接続エイリアスを削除する前に、以下の点について確認します。

- MQTT 接続エイリアスは無効になっている。
- MQTT メッセージをパブリッシュするサービスは MQTT 接続エイリアスに依存していない。
- MQTT トリガーは MQTT 接続エイリアスを使用して MQTT サーバに接続しメッセージを抽出しない。

使用中の MQTT 接続エイリアスは削除できますが、MQTT トリガーおよび MQTT メッセージパブリッシュサービスでエラーになることがあります。

MQTT 接続エイリアスを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [MQTT 設定] に移動します。
3. [MQTT 接続エイリアスの定義] リストで、エイリアスがまだ無効になっていない場合はエイリアスを無効にします。
4. エイリアスの [**削除**] 列で ✕ をクリックします。

   MQTT トリガーが MQTT 接続エイリアスを使用する場合、Integration Server ではエイリアスの削除を確認するメッセージが表示されます。

MQTT サーバへの失敗した接続の再試行間隔の指定

Integration Server と MQTT サーバ間の接続に失敗すると、Integration Server は 20 秒後に自動的に接続の再確立を試行します。watt.server.commonmessaging.connection.retryPeriod パラメータの値を変更すると、Integration Server が接続の再確立を試行する間隔を設定できます。

16 メッセージングの拡張ロギングの使用

- メッセージングの拡張ロギングについて 358
- 拡張ロギングのログエントリの内容 358
- 拡張ロギングにメッセージングログまたはサーバログを使用 361
- メッセージングの拡張ロギングの設定 361
- メッセージングログの表示 362

メッセージングの拡張ロギングについて

メッセージングの拡張ロギングは、Integration Server でメッセージの送受信時と処理時に詳細ログエントリの書き込みを指示する設定可能な機能です。拡張ロギングと呼ばれる詳細ログによって、Integration Server 内で個々のメッセージがパブリッシュされた時点からトリガーがメッセージを受信、処理、確認するまで、そのメッセージを追跡できます。

各ログメッセージはメッセージの進捗に関する情報を継続的に表し、Integration Server 内でのメッセージの進捗を追跡できます。最も重要なことは、拡張ロギングエントリには、可能な限りパブリッシュ対象または使用対象のメッセージの一意な識別子が含まれることです。個々のメッセージを追跡できると、メッセージングソリューションにおける問題のデバッグに役立ちます。

Integration Server は、拡張メッセージングロギングに対する多くの設定オプションを備えています。これらのオプションを使用して、メッセージング実装の特定分野のログエントリに焦点を当てたり、パフォーマンスに対する拡張ロギングの影響を限定したりすることができます。たとえば、拡張ロギングは Integration Server 全体ではなく、個々のメッセージング接続エイリアスに対して設定できます。エイリアスごとに、Integration Server が生成する追加ログの対象をすべてのメッセージコンシューマやプロデューサにするか、特定のメッセージコンシューマやプロデューサにするかを指定できます。また、ログレベルを設定することで、拡張メッセージングロギングを詳細にするかどうかも指定できます。

拡張ロギングは JMS メッセージングおよび webMethods メッセージングにのみ利用できます。MQTT メッセージングには利用できません。

注:
拡張ロギングは Integration Server のアクティビティ専用です。拡張ロギングには、メッセージングプロバイダで発生するアクティビティに関する情報は含まれません。

拡張ロギングのログエントリの内容

拡張ロギングのエントリには、他のログエントリに含まれる通常のタイムスタンプとログメッセージテキストに加え、ログエントリの作成対象のメッセージに関する詳細が含まれます。

こうしたメッセージ詳細には、UUID、プロバイダによって割り当てられたメッセージ ID、メッセージの送受信に使用される接続エイリアスが含まれることがあります。各ログエントリには、同じ順序でメッセージ詳細が含まれます。これはログエントリを確認するときに役立つことがあります。

ログエントリに含まれるメッセージ情報は、Integration Server がログエントリを書き込むパブリッシュまたはサブスクライブの時点によって異なります。たとえば、メッセージの送信に関するログエントリには、メッセージのプロバイダ ID は含まれません。これは、メッセージングプロバイダがまだメッセージを受信していないためです。クライアントサイドキューへのメッセージ書き込みに関するログエントリには、トリガー名が含まれません。ドキュメントに UUID が指定されていないか、パブリッシュサービスでカスタム UUID が許可されていない場合、ドキュメントを Universal Messaging にパブリッシュする準備用のログエントリには UUID が含まれません。これは、Integration Server がまだ UUID を生成していないためです。

注:

メッセージは Integration Server によってパブリッシュ、受信、処理されるため、ログエントリには、可能な限り、メッセージの追跡を簡単にするメッセージ ID または UUID が含まれます。

次の表に、送受信されたメッセージへの各拡張ロギングエントリの情報を示します。

メッセージのプロパティ	説明
UUID	送受信されるメッセージの汎用一意識別子。 ■ webMethods メッセージングでは、ドキュメントをパブリッシュするときに *_env/uuid* フィールドの値を設定できます。ユーザ定義 UUID を使用できるのは、pub.publish:publish サービスによって Universal Messaging に送信されるメッセージに対してだけです。UUID 値を設定しない場合、またはユーザ定義 UUID を設定できないサービスを使用している場合、Integration Server によってパブリッシュ時に UUID が割り当てられます。 ■ JMS メッセージングの場合、JMS メッセージの送信時に *JMSMessage/properties/uuid* フィールドに UUID を設定できます。JMS メッセージングの場合、*uuid* フィールドはオプションです。UUID を指定しない場合、Integration Server による UUID の生成または割り当ては行われません。
メッセージング接続エイリアス	送受信されたメッセージに関連付けられたメッセージング接続エイリアスの名前。 ■ webMethods メッセージングの場合、Universal Messaging 接続エイリアスです。拡張ロギングを webMethods Broker と共に使用することはできません。 ■ JMS メッセージングの場合、JMS 接続エイリアスです。
宛先	メッセージが送信された宛先。 ■ webMethods メッセージングの場合、宛先はメッセージの送信先チャネルの名前です。チャネル名は、Integration Server がインスタンスドキュメントをパブリッシュした対象のパブリッシュ可能ドキュメントタイプに対応します。 ■ JMS メッセージングの場合、宛先には JMS メッセージの送信先であるキューまたはトピックの JNDI 検索名が含まれます。
メッセージ ID	メッセージングプロバイダによりメッセージに割り当てられた ID。 ■ webMethods メッセージングの場合、メッセージ ID は Universal Messaging で最初にメッセージが受信されたときに生成される Universal Messaging イベント ID です。 ■ JMS メッセージングの場合、メッセージ ID は JMS プロバイダの代わりに生成される JMSMessageID です。

メッセージのプロパティ	説明
トリガー名	メッセージを受信したトリガーの名前。

server.log に書き込まれる拡張ロギングエントリは、専用メッセージングログに書き込まれる拡張ロギングエントリと同じではありません。server.log エントリとメッセージングログエントリで使用される形式は異なります。ただし、server.log エントリとメッセージングログエントリには、同じログメッセージテキストとメッセージプロパティが含まれます。server.log またはメッセージングログへの拡張ロギングの詳細については、361 ページの「拡張ロギングにメッセージングログまたはサーバログを使用」を参照してください。

拡張メッセージングロギングのログレベル

拡張ロギングの一部としてパブリッシュまたは受信されたメッセージに対して書き込まれるログエントリの数はログレベルによって異なります。メッセージングログに書き込まれるログエントリの場合、メッセージング監査ロガーのログレベルによってログレベルが決まります。server.log ログに書き込まれるメッセージの場合、0168 メッセージング (拡張ロギング) サーバログ機能のログレベルによってログレベルが決まります。

注:
メッセージング監査ロガーのログレベルは、サーバログの 0168 メッセージング (拡張ロギング) サーバログ機能のログレベルと異なることがあります。

メッセージングの拡張ロギングを生成するには、ログレベルは以下の表に示されるレベルのいずれか 1 つに設定される必要があります。

現在のレベル	Integration Server で記録するエントリの対象
Info	メッセージの送信、受信、処理の成功または失敗など、主要な送受信アクション。Info のログレベルは拡張メッセージングロギングの標準ログと見なされます。
Debug	プリプロセス、メッセージの受信確認など、送受信アクティビティの詳細。Debug のログレベルは拡張メッセージングロギングの詳細ログと見なされます。
Trace	追加メッセージに対するメッセージングプロバイダのポーリングの開始と終了など、送受信アクティビティのさらなる詳細。 **注:** Trace レベルメッセージは JMS メッセージングにのみ提供されます。

注:
各ログレベルには、対応するタイプのメッセージおよび上位レベルのすべてのメッセージが含まれます。たとえば、Trace レベルには Debug および Info メッセージが含まれます。

拡張ロギングにメッセージングログまたはサーバログを使用

拡張ロギングを使用するようメッセージング接続エイリアスを設定する場合、Integration Server がログエントリを server.log またはメッセージングログに書き込むよう指定できます。

メッセージングログは、Integration Server によるメッセージの送受信用に書き込まれる拡張ロギングエントリ専用の監査ログです。メッセージング監査ログの形式では、ログメッセージテキスト、メッセージプロパティなどのログエントリを列に並べます。そのため、ログが使いやすくなります。

サーバログはジャーナルログであり、すべてのサーバログ機能のログエントリを含みます。メッセージングログとサーバログは Integration Server Administrator またはテキストエディタで表示できます。

拡張ロギングエントリの宛先としてメッセージングログまたはサーバログを使用できます。ただし、同じメッセージング接続エイリアスで両方の場所に書き込むように設定することはできません。

メッセージングの拡張ロギングの設定

Integration Server は、拡張ロギングに対する多くの設定オプションを備えています。さまざまなオプションによって、ニーズに合ったログデータを Integration Server で生成できます。たとえば、特定のメッセージング接続エイリアスを使用して送信されたすべてのメッセージのログ表示が必要なことがあります。または、ログエントリを絞り込み、メッセージング接続エイリアスによって特定の宛先に送信されたメッセージのログエントリのみ表示する必要がある場合があります。

メッセージングの拡張ロギングを設定する場合、以下のことを決定する必要があります。

- 拡張ロギングを使用する対象のメッセージ接続エイリアス。拡張ロギングはメッセージング接続エイリアスごとに設定します。webMethods メッセージング用の Universal Messaging 接続エイリアスまたは JMS 接続エイリアスの拡張ロギングを設定できます。

  Universal Messaging 接続エイリアスの拡張ロギングの設定の詳細については、262 ページの「Universal Messaging 接続エイリアスの作成」を参照してください。

  JMS 接続エイリアスの拡張ロギングの設定の詳細については、307 ページの「JMS 接続エイリアスの作成」を参照してください。

- Integration Server が拡張ロギングエントリを server.log に書き込むかメッセージングログに書き込むか。

- メッセージング接続エイリアスを使用するメッセージのコンシューマとプロデューサのどちらか一方または両方に対する拡張ロギングを生成するかどうか。さらに、追加ログが必要な特定の宛先を指定することで、Integration Server で拡張ロギングを実行する対象のメッセージプロデューサをさらに制限できます。Integration Server はメッセージング接続エイリアスを使用して特定の宛先に送信されるメッセージに対してのみ拡張ロギングを書き込みます。同様に、Integration Server では、メッセージング接続エイリアスを使用してメッセージを受信するすべてのトリガー、または特定のトリガーのみに対して追加ログを生成するよう指定することもできます。

- 拡張ロギングのログレベル。Integration Server が拡張ロギングエントリをメッセージングログに書き込むかサーバログに書き込むかによって、ログレベルの設定場所が異なります。
  - メッセージングログの場合、メッセージング監査ロガーでログレベルを設定します。メッセージング監査ロガーの設定の詳細については、『*webMethods Audit Logging Guide*』を参照してください。
  - サーバログの場合、0168 メッセージング (拡張ロギング) サーバログ機能にログレベルを設定します。Integration Server で拡張ロギングエントリを書き込むには、この機能を Info ログレベル以上に設定する必要があります。server.log の設定の詳細については、237 ページの「サーバログに記録する情報の量と種類の指定」を参照してください。

拡張ロギングはパフォーマンスに影響します。Integration Server による追加ログによって、リソースが消費され、パフォーマンスに悪影響があります。追加ログエントリによって、ディスクの空き容量が急速に減少することがあります。メッセージングの拡張ロギングを使用する場合は、以上の情報に留意してください。

メッセージングログの表示

テキストエディタまたは Integration Server Administrator を使用してメッセージングログを表示できます。Integration Server Administrator では現在のメッセージングログのみ表示されることに注意してください。Integration Server Administrator には、アーカイブされたメッセージングログのログエントリは含まれません。

また、Integration Server Administrator は、特定の UUID に対応するすべてのエントリを現在のメッセージングログで検索する機能を備えています。[**ログ**] > [メッセージング] ページで、[**基本**] をクリックして、指定された UUID を含むすべてのエントリのメッセージングログを検索します。または、ログエントリを表示する UUID の行で ▼ をクリックします。

Integration Server Administrator でメッセージングログを表示する方法の詳細については、『*webMethods Audit Logging Guide*』を参照してください。

17 Universal Messaging クライアントログの設定

- Universal Messaging クライアントログ 364
- Universal Messaging クライアントログの設定 364
- イベントライフサイクルログ処理 365

Universal Messaging クライアントログ

Universal Messaging は、クライアントのアクティビティおよび Universal Messaging サーバとのやりとりを記録します。特に、Universal Messaging はクライアントの予想から逸脱しているサーバからの応答を記録します。通常これらの応答は例外の形式をとります。たとえば、クライアントにサーバへ接続する権限が与えられていない場合、サーバはセキュリティの例外を返し、Universal Messaging はその例外を記録します。Integration Server は、Universal Messaging の代わりに Integration Server Administrator 内で、このログを設定する機能を備えています。

Universal Messaging クライアントログファイルは umClient.log と呼ばれ、*Integration Server_directory* /instances/*instanceName*/logs ディレクトリに置かれます。

また、Integration Server のイベントライフサイクルログ処理も設定できます。イベントライフサイクルログ処理はトレースログ処理とも呼ばれ、パブリッシュする Universal Messaging クライアントからサブスクライブする Universal Messaging クライアントへの Universal Messaging サーバを介したメッセージ (イベント) のフローを記述する詳細ログファイルを作成します。これは、Universal Messaging クライアントログとは異なります。詳細については、365 ページの「イベントライフサイクルログ処理」を参照してください。

Universal Messaging クライアントログの設定

UM クライアントロガーを設定して Universal Messaging クライアントログファイルの詳細レベルを制御できます。

Universal Messaging クライアントログを設定するには

1. Integration Server Administrator を開きます。
2. [**ログ**] > [ログ設定] に移動します。
3. [**ロガーリスト**] で [**UM クライアントロガー**] をクリックします。
4. [**UM クライアントロガーの編集**] をクリックします。
5. [UM クライアントロガー設定] の下で、UM クライアントログを設定する次の情報を指定します。

フィールド	説明
ログレベル	ログに書き込まれる情報を指定します。各レベルはそのレベル以上のログエントリを出力します。
ログサイズ	ログファイルの最大サイズ (MB)。ログがこのサイズに達すると、Integration Server は umClient.log.*number* と呼ばれるバックアップにファイルをロールオーバ-し、新しいファイルを作成します。

フィールド	説明
ログの深さ	ログローリングを使用するときディスクに保持するバックアップログファイルの数この数に達すると、Integration Server は最も古いファイルを削除します。

6. [**変更内容の保存**] をクリックします。

7. ログサイズまたはログの深さを変更した場合は、Integration Server を再起動します。

注:
このロガーは無効にできません。

イベントライフサイクルログ処理

イベントライフサイクルログ処理はトレースログ処理とも呼ばれ、トレースロガーを使用して、メッセージ (イベント) のフローを記述する詳細ログを作成します。メッセージは Universal Messaging クライアントによってパブリッシュされ、Universal Messaging 経由でルーティングされ、サブスクライブする Universal Messaging クライアントによって受信されます。

注:
Integration Server は、Universal Messaging にメッセージをパブリッシュする場合、および通常トリガーにより Universal Messaging からメッセージを受信する場合、Universal Messaging クライアントとして機能します。

イベントライフサイクルログ処理が有効な場合、Universal Messaging サーバおよび Integration Server (Universal Messaging クライアント) のイベントアクティビティが取得されます。Integration Server および Universal Messaging サーバからのイベントライフサイクルログは、正確に同じ情報を提供するわけではありません。Integration Server のイベントライフサイクルログには、クライアントのアクティビティが記録されます。一方、Universal Messaging のイベントライフサイクルログには、メッセージングサーバのアクティビティが記録されます。両方合わせて、ログはイベントが製品間でどのように移動するかを詳細に表します。これは、欠落したメッセージを探すときに役立つことがあります。メッセージのタグ (UUID) がわかる場合、ログを調べて、メッセージが最後にシステムに記録された箇所を確認できます。また、メッセージ処理が遅い場合、イベントライフサイクルログを使用して、処理遅延が発生した場所を確認できます。

Integration Server でイベントライフサイクルログ処理を設定する場合、次の内容を指定します。

- トレースレベル。ログレベルは Trace、Info、または Off です。
- トレースストア。UM クライアントがイベントライフサイクルログ処理を実行する対象のメッセージストア。メッセージストアは、Integration Server がメッセージを送受信するチャネルまたはキューです。Integration Server がやりとりするすべてのメッセージストアのイベントライフサイクルログ処理を実行するよう、Integration Server を設定できます。
- トレースログのサイズ。各メッセージストアのイベントライフサイクルログの最大サイズ。イベントライフサイクルログがこのサイズに達すると、UM クライアントはログのアーカイブ (.zip) を作成し、ストアの新しいライフサイクルログを開始します。

- トレースフォルダのサイズ。アーカイブを含めすべてのイベントライフサイクルログを組み合わせた最大サイズ。すべてのイベントライフサイクルログおよびアーカイブの合計サイズがこの値を超える場合、UM クライアントは最も古いアーカイブ (.zip) ファイルを削除します。

ログレベルが Trace または Info である場合、Integration Server がトレースストアリストのキューまたはチャネルにメッセージを送信または受信すると、UM クライアントはストアのログファイルにライフサイクルログエントリを書き込みます。UM クライアントはストアごとに異なるライフサイクルログを使用します。すべてのライフサイクルログは次のディレクトリに置かれます。

Integration Server_directory /instances/*instanceName*/logs/umEventLifecycle

次に例を示します。

- *Integration Server_directory* /instances/*instanceName*/logs/umEventLifecycle/wm/is/my/doc

  wm/is/my/doc は、my:doc という名前のパブリッシュ可能なドキュメントタイプに対応するチャネルです。

- *Integration Server_directory* /instances/*instanceName*/logs/umEventLifecycle/queue1

以下に、イベントライフサイクルログ処理のログエントリの例を示します。

```
[2021-06-25 16:44:00,809][TRACE][Adding durable asynchronous subscriber. windowSize=-1,
 selector=null, autoAck=false,
durableInfo={durableName=J_jRQtEolDEurgAVIwwwADXaZQs_##serialtest__trig,
durableType=Named, nameId=11, selector=},
Conn={SSID:544f00000000,Local:127.0.0.1:59908,URL:nsp://127.0.0.1:9000}][nChannelImpl][Server
 Thread Pool - serialtest:trig]
[2021-06-25 16:44:00,810][INFO][Asynchronous durable subscriber successfully
added.][nChannelImpl][Server Thread Pool - serialtest:trig]
[2021-06-25 18:20:56,388][INFO][Publish request. nPublished={EID=0,
tag=wm:fe1b2bf0-d60b-11eb-8963-000000000001},
Conn={SSID:544d00000000,Local:127.0.0.1:59384,URL:nsp://127.0.0.1:9000}][nTransaction][HTTP
 Handler 127.0.0.1]
[2021-06-25 18:20:56,390][INFO][Publishing & committing transaction.
transactionId=92689689214976, ttl=1000, publishTime=-1,
Conn={SSID:544d00000000,Local:127.0.0.1:59384,URL:nsp://127.0.0.1:9000}, eventsCount=1,
 events={EID=0, tag=wm:fe1b2bf0-d60b-11eb-8963-000000000001}][nTransaction][HTTP
Handler 127.0.0.1]
[2021-06-25 18:20:56,393][TRACE][Publishing response to client. nConsumeEvent={EID=147,
 tag=wm:fe1b2bf0-d60b-11eb-8963-000000000001,
redelivery=0}][nClientChannelList][UM-Connection-Reader:3 Active]
[2021-06-25 18:20:56,393][TRACE][Received event on client. nConsumeEvent={EID=147,
tag=wm:fe1b2bf0-d60b-11eb-8963-000000000001,
redelivery=0}][nClientChannelList][UM-Connection-Reader:3 Active]
```

注:
イベントライフサイクルログで、tag は UUID と同義です。

イベントライフサイクルログ処理の設定

Integration Server でイベントライフサイクルログ処理を設定する場合、ログレベル、ログ処理を実行する対象メッセージストア (チャネルまたはキュー)、およびIntegration Server をホストするマシンでライフサイクルログ処理が使用できる容量を指定します。

Integration Server (UM クライアント) のイベントライフサイクルログ処理設定は、Universal Messaging のライフサイクルログ処理設定とは異なります。Integration Server および Universal Messaging サーバでイベントライフサイクルログ処理を設定すると、メッセージングシステムを介して進行するイベントのさらに詳しい情報を提供するログが生成されます。

イベントライフサイクルログ処理を有効にした場合、Integration Server のパフォーマンスに影響することがあります。

イベントライフサイクルログ処理を設定するには

1. Integration Server Administrator を開きます。
2. [**ログ**] > [ログ設定] に移動します。
3. [**ロガーリスト**] で [**UM クライアントロガー**] をクリックします。
4. [**UM クライアントロガーの編集**} をクリックします。
5. [UM クライアントイベントライフサイクルのログ設定] の下で、次の情報を指定して、イベントライフサイクルログ処理を有効化し設定します。

フィールド	説明
[トレースストアログのレベル]	トレースロガーのログレベル。トレースロガーはストアの詳細なログメッセージを書き込みます。以下のいずれかに設定します。 ■ [**Off**]: イベントライフサイクルログ処理はこの Integration Server に対して無効になります。これがデフォルト値です。 ■ [**Info**]: イベントライフサイクルログ処理は有効になります。トレースロガーは、[**トレースストア**] に指定されたキューまたはチャネルの情報レベルのメッセージを書き込みます。 ■ [**Trace**]: イベントライフサイクルログ処理は有効になります。トレースロガーは、[**トレースストア**] に指定されたキューまたはチャネルのトレースレベルのメッセージを書き込みます。Trace ログレベルは Info ログレベルより詳細です。
[トレースストア]	イベントライフサイクルログ処理の実行対象である、メッセージストア (具体的にはキュー名またはチャネル名) のカンマ区切りリスト。この Integration Server がメッセージを送受信するすべてのキューまたはチャネルで使用されるトレースロガーの場合、* を指定します。デフォルトは * です。 Universal Messaging のフォルダを使用するキューまたはチャネルを指定する場合、スラッシュでフォルダ名を区切ります。たとえば、パブリッシュ可能なドキュメントタイプ myFolder.mySubFolder:documentABC は、Universal Messaging

フィールド	説明
	チャネル名 wm/is/myFolder/mySubFolder/documentABC に相当します。 Integration Server では、メッセージストアのリストの妥当性を検査して、その名前のキューまたはチャネルが実際に存在するかどうかを確認することはありません。
[トレースフォルダログのサイズ]	イベントライフサイクルログ処理のすべてのログファイルの最大サイズ (MB 単位)。つまり、 *Integration Server_directory* /instances/*instanceName*/logs/umEventLifecycle にある、アーカイブを含めたすべてのログファイルの最大サイズです。デフォルトは 1024 MB に設定されています。 [**トレースフォルダログのサイズ**] の値は、各メッセージストアのトレースログ、および各メッセージストアに対して保持する多数のアーカイブログファイルを格納するのに十分な大きさに設定してください。
[トレースストアログのサイズ]	個別のメッセージストアのトレースログファイルの最大サイズ (MB 単位)。つまり、 *Integration Server_directory* /instances/*instanceName*/logs/umEventLifecycle にある個別のログファイルの最大サイズです。デフォルトは 10 MB に設定されています。 イベントライフサイクルログがこのサイズに達すると、UM クライアントはログのアーカイブ (.zip) を作成し、ストアの新しいライフサイクルログを開始します。 [**トレースフォルダログのサイズ**] の値がすべてのメッセージストアのトレースログを格納するのに十分な大きさでない場合、UM クライアントはメッセージストアのライフサイクルログのアーカイブファイルを保持しません。その代わり、UM クライアントは各ストアのトレースストアログファイルを単に循環します。

6. [**変更内容の保存**] をクリックします。

18 Web サービスに対するエンドポイントエイリアスの設定

- Web サービスエンドポイントエイリアスについて 370
- HTTP/S で使用するためのプロバイダ Web サービス記述子に対するエンドポイントエイリアスの作成 371
- HTTP/S で使用するためのコンシューマ Web サービス記述子に対するエンドポイントエイリアスの作成 379
- HTTP/S で使用するためのメッセージアドレス指定に対するエンドポイントエイリアスの作成 391
- JMS で使用するためのプロバイダ Web サービス記述子に対するエンドポイントエイリアスの作成 398
- JMS で使用するためのコンシューマ Web サービス記述子に対するエンドポイントエイリアスの作成 404
- JMS で使用するためのメッセージアドレス指定に対するエンドポイントエイリアスの作成 412
- WS セキュリティヘッダーのタイムスタンプ 418

Web サービスエンドポイントエイリアスについて

Web サービスエンドポイントエイリアスはネットワークアドレスを表しますが、オプションとして、Web サービスで使用されるセキュリティクレデンシャルを含む場合があります。ネットワークアドレスプロパティを使用して、Web サービスの動的アドレス指定を有効にできます。セキュリティクレデンシャルは、Web サービスの転送レベルとメッセージレベルの両方のセキュリティを制御するために使用できます。

Web サービス記述子では、エンドポイントエイリアスがバインダと関連付けられています。Integration Server はバインダを使用して、アドレスの定義、通信プロトコル、および特定のポートタイプのデータ形式を 1 つのコンテナに収集します。エンドポイントエイリアスとバインダとの関連付けの詳細については、『*webMethods Service Development Help*』を参照してください。

コンシューマとしての Web サービス記述子、およびこれに関連する Web サービスコネクタ (WSC) は、実行時にエイリアス情報 (アドレス情報およびセキュリティクレデンシャルを含む) を使用して、要求を生成し、Web サービスの動作を呼び出します。

プロバイダとしての Web サービス記述子は、Web サービスから WSDL ファイルが要求されたときに、エンドポイントエイリアスを使用して、address エレメント (port エレメント内にある) の location= 属性を構成します。Web サービス要求に対する応答の作成時にセキュリティクレデンシャルが使用される場合があります。

プロバイダとしての Web サービス記述子を作成する場合には、Web サービス記述子タのデフォルトバインダから表示 (および変更) して、既存のエンドポイントエイリアスを指定できます。

Integration Server では、メッセージアドレス指定エンドポイントエイリアスを使用して、要求を開始または送信したエンドポイント以外のエンドポイントに応答を送信します。つまり、WS-Addressing が有効化されている場合に、要求の SOAP メッセージに匿名ではない ReplyTo または FaultTo エンドポイントが含まれていると、Integration Server では、メッセージアドレス指定エンドポイントエイリアスを使用して、ReplyTo および FaultTo のエンドポイントに応答を送信するために使用される認証の詳細を決定します。

エンドポイントエイリアスは、一般的に次のような理由から作成されます。

- **動的エンドポイントアドレス指定**エンドポイントエイリアスを使用するとエンドポイントの実際の値が実行時に参照されるため、Web サービスを使用するたびにサーバ情報を指定または変更する手間が排除されます。
- **セキュリティ**Web サービスのプロバイダおよびコンシューマに対して WS セキュリティを設定するためにエンドポイントエイリアスが必要です。WS セキュリティ機能の実装の詳細については、『*Web Services Developer's Guide*』の WS セキュリティに関する情報を参照してください。
- **WS-Addressing**メッセージアドレス指定プロパティを使用すると、要求と異なるアドレスに応答を送信するために必要な認証クレデンシャルを指定できます。メッセージアドレス指定プロパティによって、SOAP メッセージに添付できる WS-Addressing 情報が定義されます。WS-Addressing の詳細については、『*Web Services Developer's Guide*』を参照してください。
- **WS-ReliableMessaging**信頼性の高いメッセージ処理プロパティにより、2 つのエンドポイント間 (Web サービスとクライアント、または信頼性の高いメッセージ処理ソースと宛先) で

メッセージの信頼性の高いデリバーが保証されます。信頼性の高いメッセージ処理プロパティは、個別の Web サービスエンドポイントに対して設定できます。または Integration Server で定義されるすべての Web サービスエンドポイントに対してグローバルレベルで設定することもできます。

Web サービスエンドポイントエイリアスを作成する場合は、以下の点に留意してください。

- エイリアス名は、指定された使用方法 (プロバイダまたはコンシューマ) およびプロトコル内で一意である必要があります。これにより、同じ名前を持つ複数のエンドポイントエイリアスが存在することがあります。たとえば、HTTP プロトコルに対して「aliasOne」という名前のプロバイダエイリアスがあるとします。このとき、HTTP プロトコルに対して「aliasOne」という名前のコンシューマエイリアス、および HTTPS プロトコルに対して「aliasOne」という名前のプロバイダエイリアスも存在する可能性があります。
- Integration Server では、Web サービスエンドポイントエイリアスが *Integration Server_directory* \instances*instance_name*\config\endpoints に保存されます。
- プロバイダ HTTP/S Web サービスエンドポイントエイリアスの場合は、ホスト名およびポートが必須ですが、コンシューマ HTTP/S Web サービスエンドポイントエイリアスの場合はオプションです。
- コンシューマ Web サービス記述子が常駐する Integration Server がファイアウォールの内側に設置されていて、Web サービス要求をプロキシサーバを介して送信する必要がある場合は、コンシューマ Web サービスエンドポイントエイリアスにプロキシエイリアスを割り当てることができます。
- HTTP と HTTPS 用のデフォルトプロバイダ Web サービスエンドポイントエイリアスを指定できます。プロバイダ Web サービス記述子に、デフォルトエイリアスに設定されたバインダがある場合、Integration Server は記述子用の WSDL を作成するときにデフォルトエイリアスの情報を使用します。

HTTP/S で使用するためのプロバイダ Web サービス記述子に対するエンドポイントエイリアスの作成

HTTP/S バインダを使用するプロバイダ Web サービス記述子に対して Web サービスエンドポイントエイリアスを作成する場合は、次のカテゴリの情報を指定する必要があります。

- **Web サービスエンドポイントエイリアス**エンドポイント名、説明および転送手段タイプを指定します。
- **HTTP/S 転送手段プロパティ** Web サービスが常駐するサーバを指定します。
- **WS セキュリティプロパティ**受信 SOAP 要求を復号化して確認したり、送信 SOAP 応答を暗号化して署名したりするために SOAP プロセッサで必要となる情報を指定したり、タイムスタンプ情報を追加するための詳細情報を指定します。

> **注:**
> 公開鍵や秘密鍵などの WS セキュリティクレデンシャルを Web サービスエンドポイントエイリアスで常に指定する必要はありません。この情報がエイリアスで指定されない場合、Integration Server は他の場所から情報を取得できます。WS セキュリティの認証およ

び鍵の使用方法および解決順序の詳細については、『*Web Services Developer's Guide*』を参照してください。

- **メッセージアドレス指定プロパティ**SOAP の要求および応答の WS-Addressing ヘッダーを生成するために Integration Server が使用する WS-Addressing 情報を指定します。これには、メッセージまたは障害の宛先アドレス、および受信した要求のアドレスとは異なるアドレスに応答を送信するために必要な認証クレデンシャルが含まれます。
- **信頼性の高いメッセージ処理プロパティ**Web サービスエンドポイント固有の信頼性の高いメッセージ処理の情報を指定します。デフォルトでは、Integration Server は [**設定**] > [Web サービス] > [信頼性の高いメッセージ処理] > [設定の編集] ページで定義される信頼性の高いメッセージ処理設定を、すべての Web サービスプロバイダおよびコンシューマに適用します。特定の Web サービスプロバイダまたはコンシューマのサーバレベルの信頼性の高いメッセージ処理設定を変更するには、関連する Web サービスエンドポイントエイリアスの信頼性の高いメッセージ処理プロパティを定義します。

HTTP/S で使用するための WS プロバイダ Web サービスエンドポイントエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**Web サービスエンドポイントエイリアスの作成**] をクリックします。
4. [**Web サービスエンドポイントエイリアスのプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[エイリアス]	プロバイダ Web サービスエンドポイントエイリアスの名前。 エイリアス名に次の特殊文字を使用することはできません。 # ©\ & @ ^ !% * : $ ./ \ \ ` ; , ~ + =) (| } {] [> < "
[説明]	エンドポイントエイリアスの説明。
[タイプ]	**[プロバイダ]**
[転送手段タイプ]	Web サービスへのアクセスに使用する転送プロトコルを指定します。次のいずれかを選択します。 ■ **[HTTP]** ■ **[HTTPS]**

5. [**TransportType 転送手段のプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[ホスト名または IP アドレス]	エイリアスが作成される Integration Server のホスト名または IP アドレス。 ホスト Integration Server の前面にプロキシがある場合は、プロキシサーバのホスト名または IP アドレスを指定します。
[ポート]	[**ホスト名または IP アドレス**] フィールドに指定した Integration Server に定義されている、アクティブな HTTP または HTTPS リスナーポート。 ホスト Integration Server の前面にプロキシがある場合は、プロキシサーバのポートを指定します。

6. 受信 SOAP 要求を復号化したり、送信 SOAP 要求を署名したりする必要がある場合は、[**WS セキュリティプロパティ**] で以下の手順に従います。

フィールド	指定内容
[キーストアエイリアス]	受信 SOAP 要求の復号化や送信 SOAP 応答の署名に使用する秘密鍵が格納されたキーストアのエイリアス。 **重要:** プロバイダは、対応する公開鍵をコンシューマに渡しておく必要があります。
[キーエイリアス]	要求の復号化や応答の署名に使用する秘密鍵のエイリアス。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

7. 署名済み受信 SOAP メッセージの署名用認証チェーンの妥当性検査を行う必要がある場合は、[**WS セキュリティプロパティ**] で次のように指定します。

フィールド	指定内容
[トラストストアエイリアス]	信頼関係の妥当性検査を行うために Integration Server で使用される認証局の認証のリストが格納されているトラストストアのエイリアス。

8. [**WS セキュリティプロパティ**] で、タイムスタンプを操作するときに Integration Server で使用されるタイムスタンププロパティを設定します。

フィールド	指定内容
[タイムスタンプの精度]	タイムスタンプの精度を秒にするかミリ秒にするか。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、

フィールド	指定内容
	Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。 精度の値を選択しない場合、Integration Server は watt.server.ws.security.timestampPrecisionInMilliseconds パラメータで指定された値を使用します。
[タイムスタンプの破棄までの時間 (TTL)]	送信メッセージを廃棄するまでの時間 (秒)。Integration Server では、**[タイムスタンプの破棄までの時間 (TTL)]** 値を使用して、送信メッセージの Timestamp エレメントに期限切れになるまでの時間を設定します。廃棄までの時間は、0 より大きい整数を指定する必要があります。 **[タイムスタンプの破棄までの時間 (TTL)]** の値を指定しない場合、Integration Server は watt.server.ws.security.timestampTimeToLive パラメータで指定された値を使用します。
[タイムスタンプの最大スキュー]	Web サービスクライアントとホストのクロック間で許容される差であり、タイムスタンプの期限切れの妥当性検査が成功するための最大秒数。正の整数またはゼロを指定します。 Integration Server は、WS ポリシーを介して WS セキュリティを実装する場合にのみ、タイムスタンプの最大スキュー値を使用します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。 タイムスタンプの最大スキュー値を指定しない場合、Integration Server は watt.server.ws.security.timestampMaximumSkew パラメータで指定された値を使用します。
[ユーザ名トークンの TTL]	UsernameToken の作成時刻 (wsu:Created エレメントで指定) とサーバへの到達時刻との間の許容される時間差 (秒単位) です。この制限値を超える要求はサーバによって拒否されます。デフォルト値は 300 です。
[ユーザ名トークンの未来の TTL]	wsu:Created エレメントのタイムスタンプが将来の時刻になっていることがあります。要求の作成時刻がこの設定で指定された値 (秒単位) 分だけサーバへの到達時刻を超えない場合、サーバはそのような要求を有効と見なします。デフォルト値は 60 です。

注:
[ユーザ名トークンの TTL] および **[ユーザ名トークンの未来の TTL]** の設定は watt.server.ws.security.usernameTokenTTL および watt.server.ws.security.usernameTokenFutureTTL サーバ設定プロパティを使用してグローバルレベルで設定することもできます。ただし、Web サービスエンドポイントレベルで設定がある場合、サーバはグローバルプロパティを無視します。グローバルプロパティの詳細については、1136 ページの「watt.server.」を参照してください。

WS セキュリティヘッダーのタイムスタンプの詳細については、418 ページの「WS セキュリティヘッダーのタイムスタンプ」を参照してください。

9. [**ケルベロスプロパティ**] で、以下のケルベロス関連の詳細情報を指定します。この情報は、このエンドポイントエイリアスを使用するすべてのプロバイダに使用されます。

> **注:**
> これらのフィールドは、HTTPS 転送手段タイプを使用するプロバイダエンドポイントエイリアスのみで使用できます。

フィールド	指定内容

[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキスト。
	
	以下の例で、JAAS コンテキストは `WS_KERBEROS_INBOUND` です。
	`WS_KERBEROS_INBOUND {`
	`com.sun.security.auth.module.Krb5LoginModule required`
	`refreshKrb5Config=true storeKey=true isInitiator=false debug=true;`
	`};`
	
	Integration Server で配布される is_jaas.cnf ファイルには、受信要求で使用できる IS_KERBEROS_INBOUND という名前の JAAS コンテキストが含まれます。

[プリンシパル]	ケルベロス認証で使用されるプリンシパルの名前。
[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワード。プリンシパルとそのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
[プリンシパルパスワードの再入力]	前述のプリンシパルパスワード。
[サービスプリンシパル名形式]	プリンシパルデータベースに登録されているサービスのプリンシパル名を指定する場合の形式を選択します。 **選択項目** / **目的** **[ホストベース]** サービス名とホスト名 (ホストコンピュータ) を使用してプリンシパル名を表します。 これがデフォルトです。 **[ユーザ名]** KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義されている名前付きユーザとしてプリンシパル名を表します。

フィールド	指定内容
[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスのプリンシパルの名前。この情報は、ケルベロスサービスのプロバイダからパブリッシュされた WSDL ドキュメントから取得することができます。サービスプリンシパル名は、以下の形式で指定します。 `principal-name.instance-name@realm-name`

10. [**メッセージアドレス指定プロパティ**] で、メッセージのデリバーに関する以下のアドレス指定情報を指定します。メッセージアドレス指定プロパティによって、SOAP メッセージに添付できるアドレス指定情報が定義されます。

フィールド	指定内容
[宛先]	SOAP メッセージの宛先の URI。 [**参照パラメータ**] フィールドで、メッセージがアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[応答マップ]	プロバイダが返信または障害メッセージおよび対応するメッセージアドレス指定エイリアスを送信するアドレス。Integration Server によって、アドレスにマッピングされたメッセージアドレス指定エイリアスから、応答を送信するために必要な認証の詳細が抽出されます。 [**アドレス**] フィールドで、プロバイダが返信または障害メッセージを送信する URI を指定します。 [**メッセージアドレス指定エイリアス**] リストで、Integration Server が認証の詳細を抽出するメッセージアドレス指定エンドポイントエイリアスを選択します。Integration Server によって認証の詳細が使用され、ReplyTo または FaultTo エンドポイントに応答が送信されます。 [+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

11. [**信頼性の高いメッセージ処理プロパティ**] で [**有効化**] をオンにして、作成しているエンドポイントエイリアス固有の信頼性の高いメッセージ処理情報を指定します。

12. 以下の信頼性の高いメッセージ処理情報を指定して、信頼性の高いメッセージ処理ソースと宛先間のメッセージの信頼性の高いデリバーを保証します。

フィールド	指定内容
[再伝送間隔]	信頼性の高いメッセージ処理ソースが SOAP メッセージを再送信する前に信頼性の高いメッセージ処理宛先からの確認応答を待機する間隔 (ミリ秒単位)。デフォルトは 6000 ミリ秒です。
[確認応答間隔]	信頼性の高いメッセージ処理宛先がメッセージシーケンスの確認応答を送信する前に待機する間隔 (ミリ秒単位)。指定した確認応答間隔内に受信された同じシーケンスのメッセージは、1 つのバッチで確認応答されます。確認応答間隔として指定された時間内に確認応答エンドポイントへその他のメッセージが送信されない場合、確認応答はスタンドアロンメッセージとして送信されます。 デフォルトは 3000 ミリ秒です。
[指数バックオフ]	指数バックオフアルゴリズムを使用して、確認応答のないメッセージの再送信間隔を調整するかどうかを指定します。再送信試行の間隔を調整することによって、再送信された大量のメッセージが信頼性の高いメッセージ処理宛先に集中しないようにします。

選択項目	目的
true	指定した再伝送間隔に基づいて、次の再送信までの間隔を指数的に延長します。たとえば、再伝送間隔を 2 秒と指定し、指数バックオフ値を true に設定すると、メッセージの確認応答がない状態が続いた場合、次の再送信までの間隔は 2、4、8、16、32 のようになります。これがデフォルトです。
false	すべての再送信に対して、[**再伝送間隔**] フィールドで指定されている同一の間隔を使用します。

[最大再伝送回数]	信頼性の高いメッセージ処理宛先からの確認応答を受信しなかった場合に、信頼性の高いメッセージ処理ソースがメッセージを再送信する回数。再送信の試行回数に制限がないことを指定するには、[**最大再伝送回数**] を「-1」に設定します。デフォルトは 10 です。

13. [**変更内容の保存**] をクリックします。

プロバイダ Web サービス記述子のデフォルトエンドポイントエイリアスの設定

HTTP および HTTPS プロトコルの場合、各プロトコルのデフォルトプロバイダエンドポイントエイリアスとしてプロバイダ Web サービスエンドポイントエイリアスを指定できます。デフォルトプロバイダエンドポイントエイリアスがプロバイダ Web サービス記述子のバインダに割り当てられた場合、Integration Server は記述子用の WSDL 作成時およびランタイム処理中にエイリアスの情報を使用します。プロトコルのデフォルトプロバイダエンドポイントエイリアスを変更するだけで、WSDL の生成に使用される情報およびランタイム処理に使用される情報が変更さ

れます。Web サービス記述子のバインダを編集して異なるエイリアスを指定する必要はありません。

Integration Server は次の場合にデフォルトプロバイダエンドポイントエイリアスを使用します。

- [**ポートエイリアス**] プロパティが DEFAULT(*aliasName*) に設定されたバインダまたは [**ポートエイリアス**] プロパティのエイリアスを明示的に設定していないバインダを含むプロバイダ Web サービス記述子の WSDL を作成する場合
- [**ポートエイリアス**] プロパティが DEFAULT(*aliasName*) に設定されたバインダまたは [**ポートエイリアス**] プロパティのエイリアスを明示的に設定していないバインダを含むプロバイダ Web サービス記述子のランタイム処理時
- サービスの最初のプロバイダ Web サービスのエンドポイントを作成するときに使用可能なエイリアスとして
- バインダのエンドポイントを設定するときに使用可能なエイリアスとして
- HTTP または HTTPS バインディングを使用して WSDL ドキュメントから生成された WSDL の最初のプロバイダ Web サービス記述子のバインダを作成する場合。Integration Server は転送プロトコルのデフォルトプロバイダエンドポイントエイリアスをバインダに割り当てます。Integration Server は WSDL 生成時およびランタイム処理時にエイリアスの情報を使用します。

注:
プロバイダ Web サービス記述子のバインダが [**ポートエイリアス**] プロパティの値を指定せず、バインダによって使用されるプロトコルのデフォルトプロバイダエンドポイントエイリアスがない場合、Integration Server が Web サービス記述子の WSDL ドキュメントを生成すると、Integration Server は soap:address 要素の「location=」属性を localhost:*primaryPort* に設定します。

プロバイダ Web サービス記述子と使用するデフォルトプロバイダエンドポイントエイリアスを設定するときは、以下の点に留意してください。

- プロバイダ Web サービスのみデフォルトプロバイダエンドポイントエイリアスを設定できます。
- HTTP および HTTPS プロトコルのデフォルトプロバイダエンドポイントエイリアスを設定できます。JMS のデフォルトエンドポイントエイリアスは設定できません。
- Integration Server では、デフォルトプロバイダエンドポイントエイリアスの設定を必要としません。プロトコルのデフォルトエイリアスがない場合、プロバイダ Web サービス記述子のバインダの [**ポートエイリアス**] プロパティには使用可能な値として空白行が表示されます。空白行を選択し、後でバインダによって使用されるプロトコルのデフォルトエイリアスを指定した場合、Integration Server は WSDL ドキュメントの生成時および Web サービス記述子のランタイム処理時にデフォルトプロバイダエンドポイントエイリアスの情報を使用します。つまり、プロトコルのデフォルトプロバイダエンドポイントエイリアスを 1 回設定すると、以前の空の [**ポートエイリアス**] プロパティは、そのプロトコルを使用するバインダの DEFAULT(*aliasName*) に事実上設定されます。
- デフォルトエイリアスとして使用されている Web サービスエンドポイントエイリアスを削除することはできません。

プロバイダ Web サービス記述子のデフォルトエンドポイントエイリアスを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**デフォルトプロバイダエンドポイントエイリアスの設定**] をクリックします。
4. [**デフォルトプロバイダエンドポイントエイリアスの設定**] の下で、次の 1 つ以上の処理を行います。
   - [**HTTP**] リストで、HTTP プロトコルのデフォルトエンドポイントエイリアスとして使用するエイリアスを選択します。HTTP のデフォルトエンドポイントエイリアスを設定しない場合、空白行を選択します。
   - [**HTTPS**] リストで、HTTPS プロトコルのデフォルトエンドポイントエイリアスとして使用するエイリアスを選択します。HTTPS のデフォルトエンドポイントエイリアスを設定しない場合、空白行を選択します。
5. [**更新**] をクリックします。

   Integration Server は、デフォルトプロバイダエンドポイントエイリアスを使用するバインダがあるプロバイダ Web サービス記述子の WSDL ドキュメントを更新します。

注:
デフォルトプロバイダエンドポイントエイリアスを変更した後、コンシューマは、デフォルトプロバイダエンドポイントエイリアスを使用する Web サービス記述子の WSDL ドキュメントから生成された既存の Web サービスクライアントをリフレッシュする必要があります。更新された WSDL ドキュメントを使用して Web サービスクライアントをリフレッシュすると、クライアントは変更されたエンドポイント情報を利用できます。

HTTP/S で使用するためのコンシューマ Web サービス記述子に対するエンドポイントエイリアスの作成

コンシューマ Web サービス記述子で使用する HTTP/S Web サービスエンドポイントエイリアスを作成する場合は、次のカテゴリの情報を指定する必要があります。

- **Web サービスエンドポイントエイリアス**エンドポイント名、説明および転送手段タイプを指定します。
- **HTTP/S 転送手段プロパティ** (オプション)エンドポイント URL の構成に使用するホストおよびポートを指定します。Web サービスプロバイダで転送ベースの認証が必要な場合は、これらのプロパティによって HTTP/S ヘッダーに追加する認証クレデンシャルを指定します。HTTPS 転送の場合は、これらのプロパティによって Web サービスプロバイダとの SSL 通信に使用される秘密鍵のキーストアエイリアスおよびキーエイリアスを指定します。Web サービス要求をプロキシサーバを介して送信する必要がある場合は、これらのプロパティによって Integration Server が HTTP 要求をルーティングするプロキシサーバのプロキシサーバエイリアスを指定します。

- **WS セキュリティプロパティ**Web サービスのセキュリティポリシーによって決まる、WS セキュリティヘッダーに関する情報を指定します。Web サービスのセキュリティポリシーでは、以下のような要件がある場合があります。
  - SOAP メッセージ要求にユーザ名トークンを含める。
  - SOAP メッセージ応答が復号化される。
  - SOAP メッセージ要求が署名される。
  - X.509 認証がサポートされる。
  - Timestamp エレメントがセキュリティヘッダーに追加される。

  注:
  公開鍵や秘密鍵などの WS セキュリティクレデンシャルを Web サービスエンドポイントエイリアスで常に指定する必要はありません。この情報がエイリアスで指定されない場合、Integration Server は他の場所から情報を取得できます。WS セキュリティの認証および鍵の使用方法および解決順序の詳細については、『*Web Services Developer's Guide*』を参照してください。
- **メッセージアドレス指定プロパティ**応答のデリバーに関連するアドレス指定情報を指定します。この情報には、返信の送信先の返信エンドポイント、障害の送信先を指定する障害エンドポイント、サービスに関するオプションのメタデータ (WSDL や WS ポリシーなど) が含まれます。また、メッセージを宛先にルーティングするために Integration Server によって使用される、参照パラメータと呼ばれる追加のパラメータも含まれます。
- **信頼性の高いメッセージ処理プロパティ**Web サービスエンドポイント固有の信頼性の高いメッセージ処理の情報を指定します。デフォルトでは、Integration Server は [**設定**] > [Web サービス] > [信頼性の高いメッセージ処理] > [設定の編集] ページで定義される信頼性の高いメッセージ処理設定を、すべての Web サービスプロバイダおよびコンシューマに適用します。特定の Web サービスプロバイダまたはコンシューマのサーバレベルの信頼性の高いメッセージ処理設定を変更するには、関連する Web サービスエンドポイントエイリアスの信頼性の高いメッセージ処理プロパティを定義します。

HTTP/S で使用するためのコンシューマ Web サービスエンドポイントエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**Web サービスエンドポイントエイリアスの作成**] をクリックします。
4. [**Web サービスエンドポイントエイリアスのプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[エイリアス]	プロバイダ Web サービスエンドポイントエイリアスの名前。

フィールド	指定内容
	エイリアス名に次の特殊文字を使用することはできません。 # ©\ & @ ^ !% * : $ ./ \ \ ` ; , ~ + =) (| } {] [> < "
[説明]	エンドポイントエイリアスの説明。
[タイプ]	**[コンシューマ]**
[転送手段タイプ]	Web サービスへのアクセスに使用する転送プロトコルを指定します。次のいずれかを選択します。 ■ **[HTTP]** ■ **[HTTPS]**
[実行 ACL]	この Web サービスエンドポイントエイリアスを使用できるのは、サーバ上のどのユーザグループなのかを管理する ACL。ドロップダウンリストから ACL を選択します。デフォルトでは、内部 ACL によって管理されるグループのメンバーのみが、このエイリアスを使用できます。 Web サービスコネクタまたは pub.client:soapClient サービスで、特定のエンドポイントエイリアスが *endpointAlias* 入力パラメータの値として使用されている場合に限り、Integration Server は ACL チェックを実行します。コンシューマ Web サービスのエンドポイントエイリアスが、Web サービスコネクタが使用するバインダに割り当てられている場合、Integration Server は ACL チェックを実行しません。

5. WSDL のホストやポートの情報を Web サービスエンドポイントエイリアスのホストやポートの情報で上書きする場合は、[*TransportType* **転送手段のプロパティ**]で以下の情報を指定します。Integration Server でエンドポイント URL を構成する方法の詳細については、『*webMethods Service Development Help*』を参照してください。

フィールド	指定内容
[ホスト名またはIPアドレス]	Web サービスが常駐するサーバのホスト名または IP アドレス。
[ポート番号]	**[ホスト名またはIPアドレス]** フィールドに指定したホストサーバに定義されている、アクティブな HTTP または HTTPS タイプのリスナーポート。
[認証タイプ]	コンシューマの認証に使用する認証のタイプを指定します。

選択項目	目的
[基本]	コンシューマを認証するために、基本認証を使用します。
[Digest]	コンシューマを認証するために、パスワードダイジェストを使用します。

フィールド	指定内容	
	NTLM	既にドメインにログインしているクライアントを既存のクレデンシャルを使用して認証するために、NTLM 認証を使用します。
	[ケルベロス]	転送レベルで Web サービスにケルベロス認証を使用します。このオプションを選択した場合、追加セクション [**ケルベロス転送手段のプロパティ**] がこのページに追加されます。

6. HTTPS などの転送ベースの認証に対して Web サービスエンドポイントを設定する場合は、以下のオプションのフィールドをすべてまたは一部指定します。

フィールド	指定内容
[ユーザ名]	ホストサーバで HTTPS 転送レベルでコンシューマを認証するときに使用されるユーザ名。
[パスワード]	ホストサーバでコンシューマを認証するときに使用されるパスワード。
[パスワードの再入力]	上記のパスワードを再入力します。
[キーストアエイリアス]	Web サービスホストに安全に接続するために使用される秘密鍵が格納されるキーストアに対するエイリアス。 このフィールドは、HTTPS 転送手段タイプのみで使用されます。
[キーエイリアス]	秘密鍵が格納されるキーストア内のキーに対するエイリアス。この鍵は、Web サービスホストに安全に接続するために使用されます。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。 このフィールドは、HTTPS 転送手段タイプのみで使用されます。

7. Web サービス要求をプロキシサーバを介して送信する必要がある場合は、[**プロキシエイリアス**] フィールドで、Integration Server が使用するプロキシサーバを指定するために以下のいずれかを実行します。

   - Integration Server によって特定のプロキシサーバが使用されるようにする場合は、そのプロキシサーバのエイリアスを選択します。Integration Server の [**プロキシエイリアス**] フィールドには、設定されているすべての HTTP/S および SOCKS プロキシエイリアスがリストされます。
   - Integration Server によってデフォルトのプロキシサーバが使用されるようにする場合は、このフィールドを空白のままにします。

   要求の送信時に Integration Server がプロキシサーバを使用する方法の詳細については、138 ページの「Integration Server によるプロキシサーバの使用方法」を参照してください。

8. [**ケルベロス転送手段のプロパティ**] セクションを使用すると、転送レベルのケルベロス認証を設定できます。ケルベロス関連の詳細情報を指定できます。この情報は、このエンドポイントエイリアスを使用するプロバイダによってすべてのWebサービス要求に使用されます。

> **注:**
> ケルベロス認証は HTTPS には転送レベルまたはメッセージレベルで利用できますが、HTTP には転送レベルのみ利用できます。

HTTP 転送の [**認証タイプ**] として [**ケルベロス**] を選択した場合、コンシューマエンドポイントエイリアスに対して次のフィールドが利用できます。

フィールド	説明

[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキストを指定します。
	
	以下の例で、**JAAS コンテキスト**は `KerberosClient` です。
	`KerberosClient {`
	`com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true keyTab=alice.keytab;`
	`};`

[プリンシパル]	ケルベロス認証で使用されるプリンシパルの名前を指定します。
[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワードを指定します。プリンシパルとそのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。 JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
[プリンシパルパスワードの再入力]	上記のプリンシパルパスワードを再入力します。
[サービスプリンシパル名形式]	プリンシパルデータベースに登録されているサービスのプリンシパル名を指定する場合の形式を選択します。 **選択項目** / **目的** **[ホストベース]** / サービス名とホスト名 (ホストコンピュータ) を使用してプリンシパル名を表します。 **注:** 現在、このオプションは無効になっています。Integration Server はユーザ名のみサポートしています。

フィールド	説明
[ユーザ名]	KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義されている名前付きユーザとしてプリンシパル名を表します。
[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスを指定します。この情報は、ケルベロスサービスのプロバイダからパブリッシュされた WSDL ドキュメントから取得することができます。[**サービスプリンシパル名**] は、以下の形式で指定します。 *principal-name.instance-name@realm-name*

9. このコンシューマ Web サービス記述子の WS セキュリティポリシーで、SOAP メッセージ要求にユーザ名トークンが含まれる必要がある場合は、[**WS セキュリティプロパティ**] で以下の情報を指定します。

フィールド	指定内容
[ユーザ名]	ユーザ名トークンに含めるユーザ名。
[パスワード]	ユーザ名トークンに含めるパスワード (プレーンテキストを使用)。
[パスワードの再入力]	上記のパスワードを再入力します。

10. この Web サービスで使用されるセキュリティポリシーで要求が署名されていることが必要な場合は、X.509 認証トークンを含めるか、SOAP メッセージ応答が暗号化される必要があります。次のように指定します。

フィールド	指定内容
[キーストアエイリアス]	次の目的で使用される秘密鍵が格納されるキーストアに対するエイリアス。 ■ 送信 SOAP 要求を署名する ■ 送信 SOAP 要求の X.509 認証トークンを含める ■ 受信 SOAP 応答を復号化する **重要:** このコンシューマからのメッセージを検証するには、対応する公開鍵のコピーが Web サービスプロバイダに指定されている必要があります。
[キーエイリアス]	送信 SOAP メッセージの X.509 認証トークンを署名または含めたり、受信 SOAP 応答を復号化したりするために使用される秘密鍵に対する

フィールド	指定内容
	エイリアス。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

11. [**WS セキュリティプロパティ**] で、プロバイダの認証ファイルを指定します。この認証は、送信 SOAP 要求を暗号化したり、受信 SOAP 応答の妥当性検査を行ったりするために使用されます。

フィールド	指定内容
[**パートナの認証**]	公開鍵が保存されている、プロバイダの認証のパスおよびファイル名。

12. この Web サービスコンシューマで使用されるセキュリティポリシーで、信用のある認証局によって応答の妥当性検査が行われる必要がある場合は、[**WS セキュリティプロパティ**] で次のように指定します。

フィールド	指定内容
[**パートナの認証**]	プロバイダの認証を含むファイルのパスおよびファイル名。
[**トラストストアエイリアス**]	信頼関係の妥当性検査を行うために Integration Server で使用される認証局の認証のリストが格納されているトラストストアのエイリアス。

13. [**WS セキュリティプロパティ**] で、Integration Server がセキュリティヘッダーのタイムスタンプを処理する方法を設定します。

フィールド	指定内容
[**タイムスタンプの精度**]	タイムスタンプの精度を秒にするかミリ秒にするか。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。 精度の値を選択しない場合、Integration Server は watt.server.ws.security.timestampPrecisionInMilliseconds パラメータで指定された値を使用します。
[**タイムスタンプの破棄までの時間 (TTL)**]	送信メッセージを廃棄するまでの時間 (秒)。Integration Server では、[**タイムスタンプの破棄までの時間 (TTL)**] 値を使用して、送信メッセージの Timestamp エレメントに期限切れになるまでの時間を設定します。[**タイムスタンプの破棄までの時間 (TTL)**] 値は、0 より大きい整数を指定する必要があります。

フィールド	指定内容
	廃棄までの時間を指定しない場合、Integration Server は watt.server.ws.security.timestampTimeToLive パラメータで指定された値を使用します。
[タイムスタンプの最大スキュー]	Web サービスクライアントとホストのクロック間で許容される差であり、タイムスタンプの期限切れの妥当性検査が成功するための最大秒数。正の整数またはゼロを指定します。 Integration Server は、WS ポリシーを介して WS セキュリティを実装する場合にのみ、タイムスタンプの最大スキュー値を使用します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。 タイムスタンプの最大スキュー値を指定しない場合、Integration Server は watt.server.ws.security.timestampMaximumSkew パラメータで指定された値を使用します。
[ユーザ名トークンの TTL]	UsernameToken の作成時刻 (wsu:Created エレメントで指定) とサーバへの到達時刻との間の許容される時間差 (秒単位) です。この制限値を超える要求はサーバによって拒否されます。デフォルト値は 300 です。
[ユーザ名トークンの未来の TTL]	wsu:Created エレメントのタイムスタンプが将来の時刻になっていることがあります。要求の作成時刻がこの設定で指定された値 (秒単位) 分だけサーバへの到達時刻を超えない場合、サーバはそのような要求を有効と見なします。デフォルト値は 60 です。

注:
[ユーザ名トークンの TTL] および **[ユーザ名トークンの未来の TTL]** の設定は watt.server.ws.security.usernameTokenTTL および watt.server.ws.security.usernameTokenFutureTTL サーバ設定プロパティを使用してグローバルレベルで設定することもできます。ただし、Web サービスエンドポイントレベルで設定がある場合、サーバはグローバルプロパティを無視します。グローバルプロパティの詳細については、1136 ページの「watt.server.」を参照してください。

WS セキュリティヘッダーのタイムスタンプの詳細については、418 ページの「WS セキュリティヘッダーのタイムスタンプ」を参照してください。

14. **[ケルベロスプロパティ]** で、以下のケルベロス関連の詳細情報を指定します。この情報は、このエンドポイントエイリアスを使用するすべての Web サービス要求に使用されます。

注:
これらのフィールドは、HTTPS 転送手段タイプを使用するコンシューマエンドポイントエイリアスのみで使用できます。

フィールド	指定内容

[JAAS コンテキスト]	ケルベロス認証で使用されるカスタム JAAS コンテキスト。
	以下の例で、JAAS コンテキストは `KerberosClient` です。
	`KerberosClient {`
	`com.sun.security.auth.module.Krb5LoginModule required`
	`useKeyTab=true keyTab=alice.keytab;`
	`};`
	Integration Server で配布される is_jaas.cnf ファイルには、受信要求で使用できる IS_KERBEROS_INBOUND という名前の JAAS コンテキストが含まれます。

プリンシパル	ケルベロス認証で使用されるプリンシパルの名前。
[プリンシパルパスワード]	KDC に対してプリンシパルを認証するときに使用されるプリンシパルのパスワード。プリンシパルとそのパスワードが含まれるキータブファイルを認証用に使用しない場合に、プリンシパルパスワードを指定します。パスワードは、さまざまな暗号化アルゴリズムを使用して暗号化することができます。JAAS ログインコンテキストに `useKeyTab=false` が含まれる場合は、プリンシパルパスワードを指定する必要があります。
[プリンシパルパスワードの再入力]	前述のプリンシパルパスワード。
[サービスプリンシパル名形式]	プリンシパルデータベースに登録されているサービスのプリンシパル名を指定する場合の形式を選択します。 **選択項目** / **目的** **[ホストベース]** サービス名とホスト名 (ホストコンピュータ) を使用してプリンシパル名を表します。 これがデフォルトです。 **[ユーザ名]** KDC に対する認証で使用される LDAP またはセントラルユーザディレクトリで定義されている名前付きユーザとしてプリンシパル名を表します。
[サービスプリンシパル名]	ケルベロスクライアントがアクセスするサービスのプリンシパルの名前。この情報は、ケルベロスサービスのプロバイダからパブリッシュされた WSDL ドキュメントから取得することができます。サービスプリンシパル名は、以下の形式で指定します。 `principal-name.instance-name@realm-name`

15. [**メッセージアドレス指定プロパティ**] で、Web サービスへのメッセージのデリバーに関する以下のアドレス指定情報を指定します。メッセージアドレス指定プロパティによって、SOAP メッセージに添付できるアドレス指定情報が定義されます。

フィールド	指定内容
[認識の必要性]	受信側 (ヘッダーが対象とするアクタまたは役割) が WS-Addressing ヘッダーを処理する必要があるかどうか。必須の WS-Addressing ヘッダーを処理できない受信側は、メッセージを拒否して SOAP 障害を返します。 [**認識の必要性**] によって WS-Addressing ヘッダーの `mustUnderstand` 属性が決定されます。 **選択項目** — **目的** **[True]** — WS-Addressing ヘッダーの処理が受信側で必要となることを示します。 [**認識の必要性**] で [**True**] を選択し、SOAP ノードで認識されないか処理できないヘッダーが受信されると、障害が返されます。 **[False]** — WS-Addressing ヘッダーの処理がオプションであることを示します。これがデフォルトです。 **注:** SOAP 1.1 では、mustUnderstand 属性の値は True と False ではなく 0 と 1 でした。ただし、Integration Server ではどちらの値のセットも同様に処理され、必要な変換が実行されます。 SOAP 1.1 での mustUnderstand および actor 属性の詳細については、『*Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000*』を参照してください。 SOAP 1.2 での mustUnderstand および role 属性の詳細については、『Simple Object Access Protocol (SOAP) 1.2 specification』を参照してください。
[ロール]	SOAP メッセージ内の WS-Addressing ヘッダーのターゲット。[役割] によって WS-Addressing ヘッダーの role 属性の値が決定されます。actor または role 属性によって、WS-Addressing ヘッダーエントリの受信側の URI が指定されます。 **注:** SOAP 1.1 では、role 属性は actor という名前です。ただし、Integration Server では両方の名前は同様に処理され、必要な変換が実行されます。 **選択項目** — **目的**

フィールド	指定内容
	[最終的な受信者] 受信側が SOAP メッセージの最終的な宛先であることを示します。これがデフォルトです。
	[次へ] role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/next」 ■ **SOAP 1.1 の場合:** 「http://schemas.xmlsoap.org/soap/actor/next」
	[なし] role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」 ■ **SOAP 1.1 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」
	[その他] ヘッダーのターゲットを指定します。通常、これは URI です。
[宛先]	SOAP メッセージの宛先の URI。 [**参照パラメータ**] フィールドで、要求がアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。複数の参照パラメータを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[送信者]	SOAP メッセージのソースの URI。[**アドレス**] フィールドに URI を入力します。 オプションで、[**参照パラメータ**] フィールドに、メッセージを宛先にルーティングするために必要な追加のパラメータを指定します。サービスに関するメタデータ (WSDL や WS ポリシーなど) をオプションで [**メタデータエレメント**] フィールドに指定することもできます。複数の参照パラメータおよびメタデータエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[ReplyTo]	Web サービスが応答 (返信) メッセージを送信する宛先の URI。このプロパティはオプションです。 この値を指定しない場合、この URI のデフォルト値は、Web サービス記述子に添付された WS-Addressing ポリシーによって異なります。コンシューマエンドポイントエイリアスの場合、この値のデフォルトは次のとおりです。 ■ http://www.w3.org/2005/08/addressing/anonymous (WS-Addressing の最終バージョンの場合)

フィールド	指定内容
	■ http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous (WS-Addressing のサブミットバージョンの場合) [**参照パラメータ**] フィールドで、応答メッセージがアドレス指定されるエンドポイント参照の <wsa:ReferenceParameters> プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[FaultTo]	SOAP 障害メッセージのルーティング先の URI。このプロパティはオプションです。 [**参照パラメータ**] フィールドで、障害メッセージがアドレス指定されるエンドポイント参照の <wsa:ReferenceParameters> プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

16. [**信頼性の高いメッセージ処理プロパティ**] で [**有効化**] をオンにして、作成しているエンドポイントエイリアス固有の信頼性の高いメッセージ処理情報を指定します。

17. 以下の信頼性の高いメッセージ処理情報を指定して、信頼性の高いメッセージ処理ソースと宛先間のメッセージの信頼性の高いデリバーを保証します。

フィールド	指定内容
[再伝送間隔]	信頼性の高いメッセージ処理ソースが SOAP メッセージを再送信する前に信頼性の高いメッセージ処理宛先からの確認応答を待機する間隔 (ミリ秒単位)。デフォルトは 6000 ミリ秒です。
[確認応答間隔]	信頼性の高いメッセージ処理宛先がメッセージシーケンスの確認応答を送信する前に待機する間隔 (ミリ秒単位)。指定した確認応答間隔内

フィールド	指定内容
	に受信された同じシーケンスのメッセージは、1 つのバッチで確認応答されます。確認応答間隔として指定された時間内に確認応答エンドポイントへその他のメッセージが送信されない場合、確認応答はスタンドアロンメッセージとして送信されます。 デフォルトは 3000 ミリ秒です。
[指数バックオフ]	指数バックオフアルゴリズムを使用して、確認応答のないメッセージの再送信間隔を調整するかどうかを指定します。再送信試行の間隔を調整することによって、再送信された大量のメッセージが信頼性の高いメッセージ処理宛先に集中しないようにします。

選択項目	目的
true	指定した再伝送間隔に基づいて、次の再送信までの間隔を指数的に延長します。たとえば、再伝送間隔を 2 秒と指定し、指数バックオフ値を true に設定すると、メッセージの確認応答がない状態が続いた場合、次の再送信までの間隔は 2、4、8、16、32 のようになります。これがデフォルトです。
false	すべての再送信に対して、[**再伝送間隔**] フィールドで指定されている同一の間隔を使用します。

[最大再伝送回数]	信頼性の高いメッセージ処理宛先からの確認応答を受信しなかった場合に、信頼性の高いメッセージ処理ソースがメッセージを再送信する回数。再送信の試行回数に制限がないことを指定するには、[**最大再伝送回数**] を「-1」に設定します。デフォルトは 10 です。

18. [**変更内容の保存**] をクリックします。

HTTP/S で使用するためのメッセージアドレス指定に対するエンドポイントエイリアスの作成

メッセージアドレス指定に対して HTTP/S Web サービスエンドポイントエイリアスを作成する場合は、次のカテゴリの情報を指定する必要があります。

- **Web サービスエンドポイントエイリアス**エンドポイント名、説明および転送手段タイプを指定します。
- **HTTP/S 転送手段プロパティ** Integration Server が応答の送信に使用する認証の詳細を指定します。HTTPS 転送の場合は、SOAP 応答の受信者との SSL 通信に使用される秘密鍵のキーストアエイリアスおよびキーエイリアスも指定します。

  Web サービス応答をプロキシサーバを介して送信する必要がある場合は、Integration Server が HTTP メッセージをルーティングするプロキシサーバのプロキシサーバエイリアスを指定します。

- **WS セキュリティプロパティ**Web サービスのセキュリティポリシーによって決まる、WS セキュリティヘッダーに関する情報を指定します。

  注:
  公開鍵や秘密鍵などの WS セキュリティクレデンシャルを Web サービスエンドポイントエイリアスで常に指定する必要はありません。この情報がエイリアスで指定されない場合、Integration Server は他の場所から情報を取得できます。WS セキュリティの認証および鍵の使用方法および解決順序の詳細については、『*Web Services Developer's Guide*』を参照してください。

- **メッセージアドレス指定プロパティ**応答メッセージのデリバーに関連するアドレス指定情報を指定します。これには、返信の送信先の返信エンドポイント、障害の送信先を指定する障害エンドポイント、サービスに関するオプションのメタデータ (WSDL や WS ポリシーなど) が含まれます。また、メッセージを宛先にルーティングするために Integration Server によって使用される、参照パラメータと呼ばれる追加のパラメータも含まれます。

注:
プロバイダ Web サービス記述子の Web サービスエンドポイントエイリアスが、その応答マップの一部としてメッセージアドレス指定エンドポイントエイリアスを使用している場合は、メッセージアドレス指定エンドポイントエイリアスを削除できません。

HTTP/S で使用するためのメッセージアドレス指定 Web サービスエンドポイントエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**Web サービスエンドポイントエイリアスの作成**] をクリックします。
4. [**Web サービスエンドポイントエイリアスのプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[エイリアス]	メッセージアドレス指定 Web サービスエンドポイントエイリアスの名前。 エイリアス名に次の特殊文字を使用することはできません。 # ©\ & @ ^ !% * : $ ./ \ \ ` ; , ~ + =) (| } {] [> < "
[説明]	エンドポイントエイリアスの説明。
[タイプ]	**[メッセージアドレス指定]**
[転送手段タイプ]	Web サービスへのアクセスに使用する転送プロトコルを指定します。 次のいずれかを選択します。 ■ **[HTTP]**

フィールド	指定内容
	■ [HTTPS]

5. HTTPS などの転送ベースの認証に対して Web サービスエンドポイントを設定する場合は、[*TransportType* **転送手段のプロパティ**] で、以下のオプションのフィールドをすべてまたは一部指定します。

フィールド	指定内容	
[認証タイプ]	コンシューマの認証に使用する認証のタイプを指定します。	
	選択項目	**目的**
	[基本]	コンシューマを認証するために、基本認証を使用します。
	[Digest]	コンシューマを認証するために、パスワードダイジェストを使用します。
	NTLM	既にドメインにログインしているクライアントを既存のクレデンシャルを使用して認証するために、NTLM 認証を使用します。
ユーザ名	ホストサーバで HTTP または HTTPS 転送レベルでプロバイダを認証するときに使用されるユーザ名。	
[パスワード]	ホストサーバでプロバイダを認証するときに使用されるパスワード。	
[パスワードの再入力]	上記のパスワードを再入力します。	
[キーストアエイリアス]	Web サービスホストに安全に接続するために使用される秘密鍵が格納されるキーストアに対するエイリアス。 このフィールドは、HTTPS 転送手段タイプのみで使用されます。	
[キーエイリアス]	秘密鍵が格納されるキーストア内のキーに対するエイリアス。この鍵は、Web サービスホストに安全に接続するために使用されます。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。 このフィールドは、HTTPS 転送手段タイプのみで使用されます。	

6. Web サービス応答をプロキシサーバを介して送信する必要がある場合は、[**プロキシエイリアス**] フィールドで、Integration Server が使用するプロキシサーバを指定するために以下のいずれかを実行します。

   ■ Integration Server によって特定のプロキシサーバが使用されるようにする場合は、そのプロキシサーバのエイリアスを指定します。Integration Server の [**プロキシエイリアス**]

フィールドには、設定されているすべての HTTP/S および SOCKS プロキシエイリアスがリストされます。

- Integration Server によってデフォルトのプロキシサーバが使用されるようにする場合は、このフィールドを空白のままにします。

応答の送信時に Integration Server がプロキシサーバを使用する方法の詳細については、138 ページの「Integration Server によるプロキシサーバの使用方法」を参照してください。

7. [**WS セキュリティプロパティ**] で、SOAP 応答の受信者の認証ファイルを指定します。この認証は、送信 SOAP 応答を暗号化したり、受信 SOAP 応答の妥当性検査を行ったりするために使用されます。

フィールド	指定内容
[パートナの認証]	SOAP 応答の受信者の認証のパスおよびファイル名。この応答には公開鍵が保存されています。

8. この Web サービスで使用されるセキュリティポリシーで、応答が署名されていること、X.509 認証トークンが含まれていること、または SOAP メッセージ応答が暗号化されていることが必要な場合は、[**WS セキュリティプロパティ**] で以下を指定します。

フィールド	指定内容
[キーストアエイリアス]	次の目的で使用される秘密鍵が格納されるキーストアに対するエイリアス。 ■ 送信 SOAP 応答に署名する ■ 送信 SOAP 応答の X.509 認証トークンを含める **重要:** この Web サービスからの応答メッセージを検証するには、受信者は対応する公開鍵を保持している必要があります。
[キーエイリアス]	送信 SOAP メッセージの X.509 認証トークンに署名したり、これを含めたりするために使用される秘密鍵に対するエイリアス。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

9. [**WS セキュリティプロパティ**] で、Integration Server がセキュリティヘッダーのタイムスタンプを処理する方法を設定します。

フィールド	指定内容
[タイムスタンプの精度]	タイムスタンプの精度を秒にするかミリ秒にするか。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、

フィールド	指定内容
	Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。 精度の値を選択しない場合、Integration Server は watt.server.ws.security.timestampPrecisionInMilliseconds パラメータで指定された値を使用します。
[タイムスタンプの破棄までの時間 (TTL)]	送信メッセージを廃棄するまでの時間 (秒)。Integration Server では、[**タイムスタンプの破棄までの時間 (TTL)**] 値を使用して、送信メッセージの Timestamp エレメントに期限切れになるまでの時間を設定します。[**タイムスタンプの破棄までの時間 (TTL)**] 値は、0 より大きい整数を指定する必要があります。 廃棄までの時間を指定しない場合、Integration Server は watt.server.ws.security.timestampTimeToLive パラメータで指定された値を使用します。
[タイムスタンプの最大スキュー]	Web サービスクライアントとホストのクロック間で許容される差であり、タイムスタンプの期限切れの妥当性検査が成功するための最大秒数。正の整数またはゼロを指定します。 Integration Server は、WS ポリシーを介して WS セキュリティを実装する場合にのみ、タイムスタンプの最大スキュー値を使用します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。 タイムスタンプの最大スキュー値を指定しない場合、Integration Server は watt.server.ws.security.timestampMaximumSkew パラメータで指定された値を使用します。

WS セキュリティヘッダーのタイムスタンプの詳細については、418 ページの「WS セキュリティヘッダーのタイムスタンプ」を参照してください。

10. [**メッセージアドレス指定プロパティ**] で、受信者への SOAP 応答のデリバーに関する以下のアドレス指定情報を指定します。メッセージアドレス指定プロパティによって、SOAP メッセージに添付できるアドレス指定情報が定義されます。

フィールド	指定内容
[認識の必要性]	受信側 (ヘッダーが対象とするアクタまたは役割) が WS-Addressing ヘッダーを処理する必要があるかどうか。必須の WS-Addressing ヘッダーを処理できない受信側は、メッセージを拒否して SOAP 障害を返します。 [**認識の必要性**] によって WS-Addressing ヘッダーの `mustUnderstand` 属性が決定されます。

フィールド	指定内容	
	選択項目	**目的**
	[True]	WS-Addressing ヘッダーの処理が受信側で必要となることを示します。 **[認識の必要性]** で **[True]** を選択し、SOAP ノードで認識されないか処理できないヘッダーが受信されると、障害が返されます。
	[False]	WS-Addressing ヘッダーの処理がオプションであることを示します。これがデフォルトです。

注:
SOAP 1.1 では、mustUnderstand 属性の値は True と False ではなく 0 と 1 でした。ただし、Integration Server ではどちらの値のセットも同様に処理され、必要な変換が実行されます。

SOAP 1.1 での mustUnderstand および actor 属性の詳細については、『*Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000*』を参照してください。

SOAP 1.2 での mustUnderstand および role 属性の詳細については、『Simple Object Access Protocol (SOAP) 1.2 specification』を参照してください。

[ロール] SOAP メッセージ内の WS-Addressing ヘッダーのターゲット。[役割] によって WS-Addressing ヘッダーの role 属性の値が決定されます。actor または role 属性によって、WS-Addressing ヘッダーエントリの受信側の URI が指定されます。

注:
SOAP 1.1 では、role 属性は actor という名前です。ただし、Integration Server では両方の名前は同様に処理され、必要な変換が実行されます。

選択項目	**目的**
[最終的な受信者]	受信側が SOAP メッセージの最終的な宛先であることを示します。これがデフォルトです。
[次へ]	role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/next」 ■ **SOAP 1.1 の場合:** 「http://schemas.xmlsoap.org/soap/actor/next」
[なし]	role 属性について次の URI を指定します。

フィールド	指定内容
	■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」 ■ **SOAP 1.1 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」
[その他]	ヘッダーのターゲットを指定します。通常、これは URI です。
[送信者]	SOAP 応答のソースの URI。
	[**参照パラメータ**] フィールドで、エンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[ReplyTo]	応答 (返信) メッセージのルーティング先の URI。このプロパティはオプションです。 この値を指定しない場合、この URI のデフォルト値は、Web サービス記述子に添付された WS-Addressing ポリシーによって異なります。 ■ WS-Addressing の最終バージョンの場合、[**ReplyTo**] のデフォルトは http://www.w3.org/2005/08/addressing/anonymous です。 ■ WS-Addressing のサブミットバージョンの場合、[**ReplyTo**] のデフォルトは http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous です。 [**参照パラメータ**] フィールドで、応答メッセージがアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[FaultTo]	SOAP 障害メッセージのルーティング先の URI。このプロパティはオプションです。

フィールド	指定内容
	[**参照パラメータ**] フィールドで、障害メッセージがアドレス指定されるエンドポイント参照の <wsa:ReferenceParameters> プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

11. [**変更内容の保存**] をクリックします。

JMS で使用するためのプロバイダ Web サービス記述子に対するエンドポイントエイリアスの作成

プロバイダ Web サービス記述子バインダで JMS 転送手段を指定する場合は、Web サービスエンドポイントエイリアスをバインダに割り当てる必要があります。JMS 上の SOAP を使用する Web サービス記述子の場合、プロバイダ Web サービスエンドポイントエイリアスでは以下の情報を指定します。

- 要求メッセージの JMS メッセージヘッダー情報 (デリバーモード、廃棄までの時間、応答の宛先など)。Integration Server ではこの情報を使用して、Web サービス記述子に対して生成された WSDL のバインディングエレメントを設定します。
- Web サービス記述子に対する JMS メッセージ上の SOAP を受信待機する SOAP-JMS トリガー。SOAP-JMS トリガーでは、JMS プロバイダで接続を作成するのに必要な JMS 接続情報も提供されます。Integration Server では、SOAP-JMS トリガーで提供される情報を使用して、JMS URI の大半が構成されます (Web サービス記述子により、targetService が決定されます)。WSDL ドキュメントでは、JMS URI が port エレメント内の address エレメントの location= 属性の値として出現します。
- WS セキュリティプロパティ。受信 SOAP 要求を復号化して確認したり、送信 SOAP 応答を暗号化して署名したりするために SOAP プロセッサで必要な情報を指定したり、タイムスタンプ情報を追加するための詳細情報を指定します。

> **注:**
> 公開鍵や秘密鍵などの WS セキュリティクレデンシャルを Web サービスエンドポイントエイリアスで常に指定する必要はありません。この情報がエイリアスで指定されない場合、Integration Server は他の場所から情報を取得できます。WS セキュリティの認証および鍵の使用方法および解決順序の詳細については、『*Web Services Developer's Guide*』を参照してください。

- Webサービスへのメッセージのデリバーに関するアドレス指定情報を提供するメッセージアドレス指定のプロパティ。これには、メッセージまたは障害の宛先アドレス、および受信した要求のアドレスとは異なるアドレスに応答を送信するために必要な認証クレデンシャルが含まれます。

プロバイダ Web サービス記述子で JMS バインダに対する Web サービスエンドポイントエイリアスを作成するときは、以下の点に留意してください。

- Web サービスエンドポイントエイリアスは、以下の項目と関連付けできます。
  - 既存の SOAP-JMS トリガー。
  - エンドポイントエイリアスと同時に作成する WS エンドポイントトリガー。
- Web サービスエンドポイントエイリアスで SOAP-JMS トリガーを使用し、その後、エイリアスをプロバイダ Web サービス記述子の JMS バインダに割り当てる場合、Web サービス記述子は SOAP-JMS トリガーに依存します。このため、起動時、または Web サービス記述子を含むパッケージが再ロードされるときに、Integration Server では Web サービス記述子のロードよりも前に SOAP-JMS トリガーをロードする必要があります。SOAP-JMS トリガーと Web サービス記述子が同じパッケージ内にない場合は、SOAP-JMS トリガーを含むパッケージで Web サービス記述子を含むパッケージに対するパッケージの依存性を作成する必要があります。
- エイリアスに割り当てられたSOAP-JMSトリガーの名前を変更する場合は、名前が変更されたトリガーを使用するようにエイリアスを更新する必要があります。
- 次のプロパティはオプションです。
  - **[デリバーモード]**
  - **[廃棄までの時間 (TTL)]**
  - **[優先順位]**
  - **[応答先名]**
  - **[応答先タイプ]**
- ここに示すプロパティのいずれかの値を指定しない場合 (または無効な値を指定した場合)、Integration Server はそのプロパティの情報を、Web サービスエンドポイントエイリアスを使用するプロバイダ Web サービス記述子に対して生成される WSDL ドキュメントに含めません。WSDL ドキュメントにそのプロパティの情報がない場合は、Java Message Service の規格で示されているように、Webサービスコンシューマでプロパティのデフォルト値が使用されます。

JMS で使用するためのプロバイダ Web サービスエンドポイントエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**Web サービスエンドポイントエイリアスの作成**] をクリックします。

4. [**Web サービスエンドポイントエイリアスのプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[エイリアス]	JMS プロバイダ Web サービスエンドポイントエイリアスの名前。 エイリアス名に次の特殊文字を使用することはできません。 # ©\ & @ ^ !% * : $ ./ \ \ ` ; , ~ + =) (| } {] [> < "
[説明]	エンドポイントエイリアスの説明。
[タイプ]	**[プロバイダ]**
[転送手段タイプ]	**JMS**

5. [**JMS 転送手段のプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[JMS トリガー名]	以下の場合に使用する SOAP-JMS トリガーの名前。 ■ JMS メッセージを受信する。 ■ この Web サービスエンドポイントエイリアスを使用して、任意の Web サービス記述子に JMS 接続プロパティを指定する WS エンドポイントトリガーを使用する場合は、[**WS エンドポイントトリガー**] を選択します。WS エンドポイントトリガーの詳細については、927 ページの「WS エンドポイントトリガーについて」を参照してください。
[デリバーモード]	要求メッセージのメッセージデリバーモード。このデリバーモードは、Web サービスクライアントが Web サービスの要求メッセージとして提供される JMS メッセージで指定する必要があります。
[廃棄までの時間 (TTL)]	JMS プロバイダで要求メッセージが期限切れになるまでに経過する可能性のあるミリ秒数。値 0 は、メッセージが期限切れにならないことを示します。

選択項目	目的
[PERSISTENT]	要求メッセージが永続的であることを示します。JMS プロバイダが失敗しても、メッセージは失われません。
[NON_PERSISTENT]	要求メッセージが永続的ではないことを示します。JMS プロバイダが失敗すると、メッセージが失われる可能性があります。

フィールド	指定内容
[優先順位]	メッセージの優先順位を指定します。JMS 規格では、優先順位レベルを 0～9 で定義します。0 は優先順位が最も低く、9 は優先順位が最も高いことを表します。
[応答先名]	Web サービスが応答 (返信) メッセージを送信する宛先の名前または検索名。SOAP-JMS トリガーによって使用される JMS 接続エイリアスが webMethods Brokerにネイティブ接続する場合は、名前を指定します。JMS 接続エイリアスが JNDI を使用して接続ファクトリを取得する場合は、検索名を指定します。この接続ファクトリは、JMS プロバイダに接続するために使用されます。
[応答先タイプ]	Web サービスが応答 (返信) メッセージを送信する宛先のタイプ。次の条件に当てはまる場合に、応答先タイプを指定します。 ■ エンドポイントエイリアスの割り当て先の Web サービス記述子が In-Out メッセージ交換パターンを使用する。 ■ SOAP-JMS トリガーによって指定される JMS 接続エイリアスが webMethods Brokerにネイティブ接続する。webMethods Brokerでは、キューとトピックに同じ名前を使用できます。応答の送信先を表すように [**応答先タイプ**] を指定する必要があります。

選択項目	目的
[キュー]	Webサービスが応答メッセージを特定のキューに送信することを示します。
[トピック]	Webサービスが要求メッセージを特定のトピックに送信することを示します。

6. [**JMS WSDL オプション**] で、以下の情報を指定します。

選択項目	目的
[接続ファクトリ名を含める]	接続ファクトリ名を JMS URI に含めます。
[JNDI パラメータを含める]	JNDI パラメータを JMS URI に含めます。

注:
WSDL ドキュメントでは、JMS URI が port エレメント内の address エレメントの location 属性の値として出現します。

7. 受信 SOAP 要求を復号化したり、送信 SOAP 応答を署名したりする必要がある場合は、[**WS セキュリティプロパティ**] で以下の手順に従います。

フィールド	指定内容
[キーストアエイリアス]	受信 SOAP 要求の復号化や送信 SOAP 応答の署名に使用する秘密鍵が格納されたキーストアのエイリアス。 **重要:** プロバイダは、対応する公開鍵をコンシューマに渡しておく必要があります。
[キーエイリアス]	要求の復号化や応答の署名に使用する秘密鍵のエイリアス。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

8. 署名済み受信 SOAP メッセージの署名用認証チェーンの妥当性検査を行う必要がある場合は、[**WS セキュリティプロパティ**] で次のように指定します。

フィールド	指定内容
[トラストストアエイリアス]	信頼関係の妥当性検査を行うために Integration Server で使用される認証局の認証のリストが格納されているトラストストアのエイリアス。

9. [**WS セキュリティプロパティ**] で、Integration Server がセキュリティヘッダーのタイムスタンプを処理する方法を設定します。

フィールド	指定内容
[タイムスタンプの精度]	タイムスタンプの精度を秒にするかミリ秒にするか。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。 精度の値を選択しない場合、Integration Server は watt.server.ws.security.timestampPrecisionInMilliseconds パラメータで指定された値を使用します。
[タイムスタンプの破棄までの時間 (TTL)]	送信メッセージを廃棄するまでの時間 (秒)。Integration Server では、廃棄までの時間の値を使用して、送信メッセージの Timestamp エレメントに期限切れまでの時間を設定します。[**タイムスタンプの破棄までの時間 (TTL)**] 値は、0 より大きい整数を指定する必要があります。 [**タイムスタンプの破棄までの時間 (TTL)**] の値を指定しない場合、Integration Server は watt.server.ws.security.timestampTimeToLive パラメータで指定された値を使用します。 **注:**

フィールド	指定内容
	[**タイムスタンプの破棄までの時間(TTL)**] 値は、[**JMS転送手段のプロパティ**] で指定した [**廃棄までの時間 (TTL)**] 値よりも大きくする必要があります。
[タイムスタンプの最大スキュー]	Web サービスクライアントとホストのクロック間で許容される差であり、タイムスタンプの期限切れの妥当性検査が成功するための最大秒数。正の整数またはゼロを指定します。 Integration Server は、WS ポリシーを介して WS セキュリティを実装する場合にのみ、タイムスタンプの最大スキュー値を使用します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。 タイムスタンプの最大スキュー値を指定しない場合、Integration Server は watt.server.ws.security.timestampMaximumSkew パラメータで指定された値を使用します。

10. [**メッセージアドレス指定プロパティ**] で、メッセージのデリバーに関する以下のアドレス指定情報を指定します。メッセージアドレス指定プロパティによって、SOAP メッセージに添付できるアドレス指定情報が定義されます。

フィールド	指定内容
[宛先]	SOAP メッセージの宛先の URI。 [**参照パラメータ**] フィールドで、メッセージがアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[応答マップ]	プロバイダが返信または障害メッセージおよび対応するメッセージアドレス指定エイリアスを送信するアドレス。Integration Server によって、アドレスにマッピングされたメッセージアドレス指定エイリアスから、応答を送信するために必要な認証の詳細が抽出されます。 [**アドレス**] フィールドで、プロバイダが返信または障害メッセージを送信する URI を指定します。 [**メッセージアドレス指定エイリアス**] リストで、Integration Server が認証の詳細を抽出するメッセージアドレス指定エンドポイントエイリア

フィールド	指定内容
	スを選択します。Integration Server によって認証の詳細が使用され、ReplyTo または FaultTo エンドポイントに応答が送信されます。 [+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

11. [**変更内容の保存**] をクリックします。

[**JMS トリガー名**] に WS エンドポイントトリガーを選択した場合は、エイリアスを保存すると WS エンドポイントトリガーが作成されます。WS エンドポイントトリガーを使用可能にするには、宛先を指定して有効化する必要があります。また、一部のメッセージ処理プロパティの値を変更することもできます。WS エンドポイントトリガーの編集の詳細については、928 ページの「WS エンドポイントトリガーの編集」を参照してください。

注:
JMS で使用するプロバイダ Web サービスエンドポイントエイリアスで WS エンドポイントトリガーが指定されている場合は、エイリアスを削除すると WS エンドポイントトリガーも削除されます。

JMS で使用するためのコンシューマ Web サービス記述子に対するエンドポイントエイリアスの作成

JMS バインダのあるコンシューマ Web サービス記述子で使用する Web サービスエンドポイントエイリアスでは、Web サービスコネクタの実行時に Integration Server が要求メッセージを送信する方法および送信先を指定します。

コンシューマ Web サービス記述子を作成するときに、Integration Server は WSDL ドキュメントから JMS 情報を抽出して、バインダ情報と共に Web サービス記述子に保存します。ただし、SOAP over Java Message Service の規格に示されているように、WSDL に必要な JMS 情報は検索バリアントおよび宛先名のみです。このため、JMS プロバイダへの接続に必要な一部の情報が WSDL から欠落している可能性があります。Integration Server は、WSDL ドキュメントで指定される JMS 情報を置き換えたり補完したりするために、JMS コンシューマ Web サービスエンドポイントエイリアス内の情報を使用します。

コンシューマ Web サービス記述子を作成するときに、メッセージアドレス指定プロパティによって、SOAP メッセージに添付できる WS-Addressing ヘッダー情報が定義されます。

JMS 上の SOAP バインドと共にコンシューマ Web サービス記述子で使用するために Web サービスエンドポイントエイリアスを作成するときは、以下の点に留意してください。

- JMS コンシューマ Web サービスエンドポイントエイリアスでは、JMS プロバイダに接続するために、次のいずれかのオプションを指定できます。
  - JNDI プロバイダエイリアスおよび接続ファクトリ
  - JMS 接続エイリアス

JMS プロバイダへの接続に関する情報がコンシューマ Web サービス記述子の作成に使用される WSDL ドキュメントに含まれていない場合、または WSDL ドキュメントに含まれる接続情報を上書きする場合、JNDI プロバイダエイリアスおよび接続ファクトリ、または JMS 接続エイリアスのみを指定します。

注:
JMS 接続エイリアスを使用して JMS プロバイダに接続すると、パフォーマンスが向上することがあります。JMS 接続エイリアスは、JNDI を使用して接続ファクトリを取得してから、webMethods Brokerにネイティブ接続することで接続を確立し、JMS プロバイダに接続できることに注意してください。

- エイリアスを割り当てたWebサービス記述子でクライアントサイドキューを使用する場合、JMS プロバイダに接続する方法として JMS 接続エイリアスを指定する必要があります。
- JMS コンシューマ Web サービスエンドポイントエイリアス内の情報は、WSDL から取得した JMS URI 情報を補完したり置き換えたりできます。
- エンドポイントエイリアスを使用して、Webサービスのセキュリティポリシーによって決まる、WS セキュリティヘッダーに関する情報を指定できます。Web サービスのセキュリティポリシーでは、以下のような要件がある場合があります。
  - SOAP メッセージ要求にユーザ名トークンを含める。
  - SOAP メッセージ応答が復号化される。
  - SOAP メッセージ要求が署名される。
  - X.509 認証。
  - Timestamp エレメントがセキュリティヘッダーに追加される。

注:
公開鍵や秘密鍵などの WS セキュリティクレデンシャルを Web サービスエンドポイントエイリアスで常に指定する必要はありません。この情報がエイリアスで指定されない場合、Integration Server は他の場所から情報を取得できます。WS セキュリティの認証および鍵の使用方法および解決順序の詳細については、『*Web Services Developer's Guide*』を参照してください。

JMS で使用するためのコンシューマ Web サービスエンドポイントエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**Web サービスエンドポイントエイリアスの作成**] をクリックします。
4. [**Web サービスエンドポイントエイリアスのプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[エイリアス]	JMS コンシューマ Web サービスエンドポイントエイリアスの名前。 エイリアス名に次の特殊文字を使用することはできません。 # ©\ & @ ^ !% * : $ ./ \ \ ` ; , ~ + =) (| } {] [> < "
[説明]	エンドポイントエイリアスの説明。
[タイプ]	**[コンシューマ]**
[転送手段タイプ]	**JMS**
[実行 ACL]	この Web サービスエンドポイントエイリアスを使用できるのは、サーバ上のどのユーザグループなのかを管理する ACL。ドロップダウンリストから ACL を選択します。デフォルトでは、内部 ACL によって管理されるグループのメンバーのみが、このエイリアスを使用できます。

5. 接続ファクトリを使用して JMS プロバイダに接続する場合は、[**JMS 転送手段のプロパティ**] で以下の手順に従います。

フィールド	指定内容
[接続に使用]	**[JNDI プロパティ]**
[JNDI プロバイダエイリアス]	管理対象オブジェクトを検索するために Integration Server で使用される JNDI プロバイダのエイリアス。JNDI プロバイダエイリアスの作成の詳細については、299 ページの「JNDI プロバイダエイリアスの作成」を参照してください。
[接続ファクトリ名]	JMS プロバイダへの接続の作成に使用する接続ファクトリの検索名。 **注:** コンシューマ Web サービス記述子の作成に使用される WSDL ドキュメントで接続ファクトリを指定しなかった場合、または接続ファクトリを上書きする場合にのみ、接続ファクトリを指定する必要があります。

6. JMS 接続エイリアスを使用して JMS プロバイダに接続する場合は、[**JMS 転送手段のプロパティ**] で以下の手順に従います。

フィールド	指定内容
[接続に使用]	**[JMS 接続エイリアス]**
[JMS 接続エイリアス]	Integration Server で JMS プロバイダへの接続に使用される JMS 接続エイリアスの名前。JMS 接続エイリアスの作成については、307 ページの「JMS 接続エイリアスの作成」を参照してください。

7. このコンシューマ Web サービス記述子の WS セキュリティポリシーで SOAP メッセージ要求にユーザ名トークンが含まれる必要がある場合は、[**WS セキュリティプロパティ**] で以下の情報を指定します。

フィールド	指定内容
[ユーザ名]	ユーザ名トークンに含めるユーザ名。
[パスワード]	ユーザ名トークンに含めるパスワード (プレーンテキストを使用)。
[パスワードの再入力]	上記のパスワードを再入力します。

8. この Web サービスで使用されるセキュリティポリシーで要求が署名されていることが必要な場合は、X.509 認証トークンを含めるか、SOAP メッセージ応答が暗号化される必要があります。次のように指定します。

フィールド	指定内容
[キーストアエイリアス]	次の目的で使用される秘密鍵が格納されるキーストアに対するエイリアス。 ■ 送信 SOAP 要求を署名する ■ 送信 SOAP 要求の X.509 認証トークンを含める ■ 受信 SOAP 応答を復号化する **重要:** このコンシューマからのメッセージを検証するには、対応する公開鍵のコピーが Web サービスプロバイダに指定されている必要があります。
[キーエイリアス]	送信 SOAP メッセージの X.509 認証トークンを署名または含めたり、受信 SOAP 応答を復号化したりするために使用される秘密鍵に対するエイリアス。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

9. [**WS セキュリティプロパティ**] で、プロバイダの認証ファイルを指定します。この認証は、送信 SOAP 要求を暗号化したり、受信 SOAP 応答の妥当性検査を行ったりするために使用されます。

フィールド	指定内容
[パートナの認証]	公開鍵が保存されている、プロバイダの認証のパスおよびファイル名。

10. この Web サービスコンシューマで使用されるセキュリティポリシーで、信用のある認証局によって応答の妥当性検査が行われる必要がある場合は、[**WS セキュリティプロパティ**] で次のように指定します。

フィールド	指定内容
[パートナの認証]	プロバイダの認証を含むファイルのパスおよびファイル名。
[トラストストアエイリアス]	信頼関係の妥当性検査を行うために Integration Server で使用される認証局の認証のリストが格納されているトラストストアのエイリアス。

11. [**WS セキュリティプロパティ**] で、Integration Server がセキュリティヘッダーのタイムスタンプを処理する方法を設定します。

フィールド	指定内容
[タイムスタンプの精度]	タイムスタンプの精度を秒にするかミリ秒にするか。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。 精度の値を選択しない場合、Integration Server は watt.server.ws.security.timestampPrecisionInMilliseconds パラメータで指定された値を使用します。
[タイムスタンプの破棄までの時間 (TTL)]	送信メッセージを廃棄するまでの時間 (秒)。Integration Server では、[**タイムスタンプの破棄までの時間 (TTL)**] 値を使用して、送信メッセージの Timestamp エレメントに期限切れになるまでの時間を設定します。[**タイムスタンプの破棄までの時間 (TTL)**] 値は、0 より大きい整数を指定する必要があります。 廃棄までの時間を指定しない場合、Integration Server は watt.server.ws.security.timestampTimeToLive パラメータで指定された値を使用します。
[タイムスタンプの最大スキュー]	Web サービスクライアントとホストのクロック間で許容される差であり、タイムスタンプの期限切れの妥当性検査が成功するための最大秒数。正の整数またはゼロを指定します。 Integration Server は、WS ポリシーを介して WS セキュリティを実装する場合にのみ、タイムスタンプの最大スキュー値を使用します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。 タイムスタンプの最大スキュー値を指定しない場合、Integration Server は watt.server.ws.security.timestampMaximumSkew パラメータで指定された値を使用します。

WS セキュリティヘッダーのタイムスタンプの詳細については、418 ページの「WS セキュリティヘッダーのタイムスタンプ」を参照してください。

12. [**メッセージアドレス指定プロパティ**] で、Web サービスへのメッセージのデリバーに関する以下のアドレス指定情報を指定します。

フィールド	指定内容
[認識の必要性]	受信側 (ヘッダーが対象とするアクタまたは役割) が WS-Addressing ヘッダーを処理する必要があるかどうか。必須の WS-Addressing ヘッダーを処理できない受信側は、メッセージを拒否して SOAP 障害を返します。 [**認識の必要性**] によって WS-Addressing ヘッダーの `mustUnderstand` 属性が決定されます。

選択項目	目的
[True]	受信側 (ヘッダーが対象とするアクタまたは役割) で、WS-Addressing ヘッダーの処理が必要となることを示します。 [**認識の必要性**] で [**True**] を選択し、SOAP ノードで認識されないか処理できないヘッダーが受信されると、障害が返されます。
[False]	WS-Addressing ヘッダーの処理がオプションであることを示します。これがデフォルトです。

注:
SOAP 1.1 では、mustUnderstand 属性の値は True と False ではなく 0 と 1 でした。ただし、Integration Server ではどちらの値のセットも同様に処理され、必要な変換が実行されます。

SOAP 1.1 での mustUnderstand および actor 属性の詳細については、『*Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000*』を参照してください。

SOAP 1.2 での mustUnderstand および role 属性の詳細については、『Simple Object Access Protocol (SOAP) 1.2 specification』を参照してください。

[ロール]	SOAP メッセージ内の WS-Addressing ヘッダーのターゲット。[役割] によって WS-Addressing ヘッダーの role 属性の値が決定されます。actor または role 属性によって、WS-Addressing ヘッダーエントリの受信側の URI が指定されます。

注:

フィールド	指定内容
	SOAP 1.1 では、role 属性は actor という名前です。ただし、Integration Server では両方の名前は同様に処理され、必要な変換が実行されます。

選択項目	目的
[最終的な受信者]	受信側が SOAP メッセージの最終的な宛先であることを示します。これがデフォルトです。
[次へ]	role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/next」 ■ **SOAP 1.1 の場合:** 「http://schemas.xmlsoap.org/soap/actor/next」
[なし]	role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」 ■ **SOAP 1.1 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」
[その他]	ヘッダーのターゲットを指定します。通常、これは URI です。

[宛先]	SOAP 要求の宛先の URI。 [**参照パラメータ**] フィールドで、要求がアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。複数の参照パラメータを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[送信者]	SOAP メッセージのソースの URI。 [**参照パラメータ**] フィールドで、エンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[ReplyTo]	応答 (返信) メッセージのルーティング先の URI。このプロパティはオプションです。

フィールド	指定内容
	この値を指定しない場合、この URI のデフォルト値は、Web サービス記述子に添付された WS-Addressing ポリシーによって異なります。 ■ WS-Addressing の最終バージョンの場合、[**ReplyTo**] のデフォルトは http://www.w3.org/2005/08/addressing/anonymous です。 ■ WS-Addressing のサブミットバージョンの場合、[**ReplyTo**] のデフォルトは http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous です。 [**参照パラメータ**] フィールドで、応答メッセージがアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[**FaultTo**]	SOAP 障害メッセージのルーティング先の URI。このプロパティはオプションです。 [**参照パラメータ**] フィールドで、障害メッセージがアドレス指定されるエンドポイント参照の `<wsa:ReferenceParameters>` プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

13. [**変更内容の保存**] をクリックします。

JMS で使用するためのメッセージアドレス指定に対するエンドポイントエイリアスの作成

JMS バインダのある Web サービス記述子で使用するメッセージアドレス指定 Web サービスエンドポイントエイリアスでは、受信者への SOAP 応答のデリバーに関するアドレス指定情報を指定します。

JMS 上の SOAP バインドと共にメッセージアドレス指定 Web サービス記述子で使用するために Web サービスエンドポイントエイリアスを作成するときは、以下の点に留意してください。

- JMS メッセージアドレス指定 Web サービスエンドポイントエイリアスでは、JMS プロバイダに接続するために、次のいずれかのオプションを指定できます。
  - JNDI プロバイダエイリアスおよび接続ファクトリ
  - JMS 接続エイリアス

  JMS プロバイダへの接続に関する情報がコンシューマ Web サービス記述子の作成に使用される WSDL ドキュメントに含まれていない場合、または WSDL ドキュメントに含まれる接続情報を上書きする場合、JNDI プロバイダエイリアスおよび接続ファクトリ、または JMS 接続エイリアスのみを指定します。

  注:
  JMS 接続エイリアスを使用して JMS プロバイダに接続すると、パフォーマンスが向上することがあります。JMS 接続エイリアスは、JNDI を使用して接続ファクトリを取得してから、webMethods Brokerにネイティブ接続することで接続を確立し、JMS プロバイダに接続できることに注意してください。
- エンドポイントエイリアスを使用して、Web サービスのセキュリティポリシーによって決まる、WS セキュリティヘッダーに関する情報を指定できます。

  注:
  公開鍵や秘密鍵などの WS セキュリティクレデンシャルをメッセージアドレス指定 Web サービスエンドポイントエイリアスで常に指定する必要はありません。この情報がエイリアスで指定されない場合、Integration Server は他の場所から情報を取得できます。WS セキュリティの認証および鍵の使用方法および解決順序の詳細については、『*Web Services Developer's Guide*』を参照してください。
- エイリアスを割り当てた Web サービス記述子でクライアントサイドキューを使用する場合、JMS プロバイダに接続する方法として JMS 接続エイリアスを指定する必要があります。

注:
プロバイダ Web サービス記述子の Web サービスエンドポイントエイリアスが、その応答マップの一部としてメッセージアドレス指定エンドポイントエイリアスを使用している場合は、メッセージアドレス指定エンドポイントエイリアスを削除できません。

» JMS で使用するためのメッセージアドレス指定 Web サービスエンドポイントエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**Web サービスエンドポイントエイリアスの作成**] をクリックします。
4. [**Web サービスエンドポイントエイリアスのプロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[エイリアス]	JMS メッセージアドレス指定 Web サービスエンドポイントエイリアスの名前。 エイリアス名に次の特殊文字を使用することはできません。 # ©\ & @ ^ !% * : $ ./ \ \ ` ; , ~ + =) (| } {] [> < "
[説明]	エンドポイントエイリアスの説明。
[タイプ]	**[メッセージアドレス指定]**
[転送手段タイプ]	**JMS**

5. 接続ファクトリを使用して JMS プロバイダに接続する場合は、[**JMS 転送手段のプロパティ**] で以下の手順に従います。

フィールド	指定内容
[接続に使用]	**[JNDI プロパティ]**
[JNDI プロバイダエイリアス]	管理対象オブジェクトを検索するために Integration Server で使用される JNDI プロバイダのエイリアス。JNDI プロバイダエイリアスの作成の詳細については、299 ページの「JNDI プロバイダエイリアスの作成」を参照してください。
[接続ファクトリ名]	JMS プロバイダへの接続の作成に使用する接続ファクトリの検索名。

6. JMS 接続エイリアスを使用して JMS プロバイダに接続する場合は、[**JMS 転送手段のプロパティ**] で以下の手順に従います。

フィールド	指定内容
[接続に使用]	**[JMS 接続エイリアス]**
[JMS 接続エイリアス]	Integration Server で JMS プロバイダへの接続に使用される JMS 接続エイリアスの名前。JMS 接続エイリアスの作成については、307 ページの「JMS 接続エイリアスの作成」を参照してください。

7. [**WS セキュリティプロパティ**] で、SOAP 応答の受信者の認証ファイルを指定します。この認証は、送信 SOAP 応答を暗号化するために使用されます。

フィールド	指定内容
[パートナの認証]	SOAP 応答の受信者の認証ファイルのパスおよびファイル名。この応答には公開鍵が保存されています。

8. この Web サービスで使用されるセキュリティポリシーで、応答が署名されていること、X.509 認証トークンが含まれていること、または SOAP メッセージ応答が暗号化されていることが必要な場合は、[**WS セキュリティプロパティ**] で以下を指定します。

フィールド	指定内容
[キーストアエイリアス]	次の目的で使用される秘密鍵が格納されるキーストアに対するエイリアス。 ■ 送信 SOAP 応答に署名する ■ 送信 SOAP 応答の X.509 認証トークンを含める **重要:** この Web サービスからの応答メッセージを検証するには、受信者は対応する公開鍵を保持している必要があります。
[キーエイリアス]	送信 SOAP メッセージの X.509 認証トークンに署名したり、これを含めたりするために使用される秘密鍵に対するエイリアス。この鍵は、[**キーストアエイリアス**] に指定されたキーストア内に存在する必要があります。

9. [**WS セキュリティプロパティ**] で、Integration Server がセキュリティヘッダーのタイムスタンプを処理する方法を設定します。

フィールド	指定内容
[タイムスタンプの精度]	タイムスタンプの精度を秒にするかミリ秒にするか。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。 精度の値を選択しない場合、Integration Server は watt.server.ws.security.timestampPrecisionInMilliseconds パラメータで指定された値を使用します。
[タイムスタンプの破棄までの時間 (TTL)]	送信メッセージを廃棄するまでの時間 (秒)。Integration Server では、[**タイムスタンプの破棄までの時間 (TTL)**] 値を使用して、送信メッセージの Timestamp エレメントに期限切れになるまでの時間を設定しま

フィールド	指定内容
	す。[**タイムスタンプの破棄までの時間 (TTL)**] 値は、0 より大きい整数を指定する必要があります。 廃棄までの時間を指定しない場合、Integration Server は watt.server.ws.security.timestampTimeToLive パラメータで指定された値を使用します。
[タイムスタンプの最大スキュー]	Web サービスクライアントとホストのクロック間で許容される差であり、タイムスタンプの期限切れの妥当性検査が成功するための最大秒数。正の整数またはゼロを指定します。 Integration Server は、WS ポリシーを介して WS セキュリティを実装する場合にのみ、タイムスタンプの最大スキュー値を使用します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。 タイムスタンプの最大スキュー値を指定しない場合、Integration Server は watt.server.ws.security.timestampMaximumSkew パラメータで指定された値を使用します。

WS セキュリティヘッダーのタイムスタンプの詳細については、418 ページの「WS セキュリティヘッダーのタイムスタンプ」を参照してください。

10. [**メッセージアドレス指定プロパティ**] で、受信者への応答 SOAP メッセージのデリバーに関する以下のアドレス指定情報を指定します。メッセージアドレス指定プロパティによって、SOAP メッセージに添付できるアドレス指定情報が定義されます。

フィールド	指定内容
[認識の必要性]	受信側 (ヘッダーが対象とするアクタまたは役割) が WS-Addressing ヘッダーを処理する必要があるかどうか。必須の WS-Addressing ヘッダーを処理できない受信側は、メッセージを拒否して SOAP 障害を返します。 [**認識の必要性**] によって WS-Addressing ヘッダーの `mustUnderstand` 属性が決定されます。

選択項目	目的
[True]	WS-Addressing ヘッダーの処理が受信側で必要となることを示します。 [**認識の必要性**] で [**True**] を選択し、SOAP ノードで認識されないか処理できないヘッダーが受信されると、障害が返されます。
[False]	WS-Addressing ヘッダーの処理がオプションであることを示します。これがデフォルトです。

フィールド	指定内容
	注: SOAP 1.1 では、mustUnderstand 属性の値は True と False ではなく 0 と 1 でした。ただし、Integration Server ではどちらの値のセットも同様に処理され、必要な変換が実行されます。 SOAP 1.1 での mustUnderstand および actor 属性の詳細については、『*Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000*』を参照してください。 SOAP 1.2 での mustUnderstand および role 属性の詳細については、『Simple Object Access Protocol (SOAP) 1.2 specification』を参照してください。
[ロール]	SOAP メッセージ内の WS-Addressing ヘッダーのターゲット。[役割] によって WS-Addressing ヘッダーの role 属性の値が決定されます。actor または role 属性によって、WS-Addressing ヘッダーエントリの受信側の URI が指定されます。 **注:** SOAP 1.1 では、role 属性は actor という名前です。ただし、Integration Server では両方の名前は同様に処理され、必要な変換が実行されます。

選択項目	目的
[最終的な受信者]	受信側が SOAP メッセージの最終的な宛先であることを示します。これがデフォルトです。
[次へ]	role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/next」 ■ **SOAP 1.1 の場合:** 「http://schemas.xmlsoap.org/soap/actor/next」
[なし]	role 属性について次の URI を指定します。 ■ **SOAP 1.2 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」 ■ **SOAP 1.1 の場合:** 「http://www.w3.org/2003/05/soap-envelope/role/none」
[その他]	ヘッダーのターゲットを指定します。通常、これは URI です。

[送信者]	SOAP 応答のソースの URI。

フィールド	指定内容
	[**参照パラメータ**] フィールドで、エンドポイント参照の <wsa:ReferenceParameters> プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[ReplyTo]	応答 (返信) メッセージのルーティング先の URI。このプロパティはオプションです。 この値を指定しない場合、この URI のデフォルト値は、Web サービス記述子に添付された WS-Addressing ポリシーによって異なります。 ■ WS-Addressing の最終バージョンの場合、[**ReplyTo**] のデフォルトは http://www.w3.org/2005/08/addressing/anonymous です。 ■ WS-Addressing のサブミットバージョンの場合、[**ReplyTo**] のデフォルトは http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous です。 [**参照パラメータ**] フィールドで、応答メッセージがアドレス指定されるエンドポイント参照の <wsa:ReferenceParameters> プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。 複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。
[FaultTo]	SOAP 障害メッセージのルーティング先の URI。このプロパティはオプションです。 [**参照パラメータ**] フィールドで、障害メッセージがアドレス指定されるエンドポイント参照の <wsa:ReferenceParameters> プロパティに対応する追加のパラメータ (存在する場合) を指定します。オプションで、サービスに関するメタデータ (WSDL や WS ポリシーなど) を [**メタデータエレメント**] フィールドに指定できます。また、[**拡張可能なエレメント**] を指定することもできます。これは、[**メタデータ**] および [**参照パラメータ**] の一部として指定した以外のエレメントです。複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

フィールド	指定内容
	複数の参照パラメータ、メタデータエレメントまたは拡張可能なエレメントを指定できます。[+] アイコンをクリックして行を追加し、[x] アイコンをクリックして行を削除します。

11. [**変更内容の保存**] をクリックします。

WS セキュリティヘッダーのタイムスタンプ

WS セキュリティヘッダーには、Timestamp エレメントおよびトークンを含めることができます。Integration Server では、送信または受信メッセージが期限切れであるかどうかを指定または検出するために、次のようにタイムスタンプを使用します。

- 送信メッセージでは、Web サービス記述子に付加される WS セキュリティポリシーに <sp:IncludeTimestamp/> アサーションが含まれる場合は、Integration Server によって作成時刻および期限切れ時刻が含まれている Timestamp エレメントがセキュリティヘッダーに追加されます。
- 受信メッセージでは、メッセージに Timestamp トークンが含まれる場合は、Timestamp トークンに基づき、Integration Server によってメッセージが期限切れ後に到着したかどうかが確認されます。

Web サービスエンドポイントエイリアスでは、メッセージのタイムスタンプの精度、メッセージの廃棄までの時間、および送信側マシンと受信側マシンのクロックの差異を考慮するかどうかを指定できます。

19 Integration Server での信頼性の高いメッセージ処理の設定

- 信頼性の高いメッセージ処理の概要 420
- Integration Server での信頼性の高いメッセージ処理の使用 421
- Integration Server での信頼性の高いメッセージ処理の設定 423
- 信頼性の高いメッセージ処理のシーケンスレポート 426
- シーケンスを閉じる 427
- シーケンスの終了 428
- 確認応答要求の送信 428

信頼性の高いメッセージ処理の概要

Integration Server では、Web Services Reliable Messaging (WS-ReliableMessaging) プロトコルを使用して、Web サービスのプロバイダとコンシューマ間で SOAP メッセージを確実に送信します。信頼性の高いメッセージ送信により、宛先エンドポイントが一時的に使用できない場合やネットワーク接続が失敗した場合でも、SOAP メッセージはデリバーされます。

以下の手順は、信頼性の高いメッセージ処理が使用される SOAP メッセージ交換の仕組みを示しています。

1. 信頼性の高いメッセージ処理ソースは、通信リンクを介して 1 つまたは一連のメッセージを信頼性の高いメッセージ処理宛先に送信します。信頼性の高いメッセージ処理ソースは、メッセージと共にメッセージの確認応答要求も受信側に送信します。
2. 信頼性の高いメッセージ処理宛先がメッセージを受信すると、メッセージごとに個別に、または一連のメッセージの 1 つの確認応答として、信頼性の高いメッセージ処理ソースに確認応答を返信します。
3. 最初の試行でメッセージがデリバーされない場合、信頼性の高いメッセージ処理ソースは、信頼性の高いメッセージ処理の設定に基づいて、メッセージを再送信します。このことは、メッセージがデリバーされて確認応答が受信されるか、シーケンスがタイムアウトまたは終了するまで行われます。

信頼性の高いメッセージ処理の用語について

Integration Server で信頼性の高いメッセージ処理を設定する前に、以下の用語を理解しておくと便利です。

- **信頼性の高いメッセージ処理ソース**SOAP メッセージを信頼性の高いメッセージ処理宛先に送信する Web サービスエンドポイント。Integration Server の場合、要求を送信するコンシューマ Web サービス記述子、または応答を送信するプロバイダ Web サービス記述子が信頼性の高いメッセージ処理ソースです。
- **信頼性の高いメッセージ処理宛先**信頼性の高いメッセージ処理ソースから確実に送信される SOAP メッセージを受信する Web サービスエンドポイント。Integration Server の場合、応答を受信するコンシューマ Web サービス記述子、または要求を受信するプロバイダ Web サービス記述子が信頼性の高いメッセージ処理宛先です。
- **信頼性の高いメッセージ処理サーバ** 信頼性の高いメッセージ処理のシナリオで Web サービスプロバイダとして機能する Integration Server。
- **信頼性の高いメッセージ処理クライアント** 信頼性の高いメッセージ処理のシナリオで Web サービスコンシューマとして機能する Integration Server。
- **メッセージシーケンス**同じ宛先を持つ 1 つのメッセージまたは一連のメッセージ。メッセージをメッセージシーケンスにグループ化することで、メッセージ送信の追跡と管理が容易になります。
- **シーケンスキー**メッセージシーケンスを識別するためのユーザ定義キー。信頼性の高いメッセージ処理クライアントは、メッセージシーケンスの送信先のエンドポイント URL 基づいてシーケンスキーをメッセージシーケンスに関連付けます。同じエンドポイント URL に送

信される複数のメッセージシーケンスがある場合、カスタムのシーケンスキーを指定して各シーケンスを識別できます。各シーケンスは、エンドポイント URL およびユーザが指定したシーケンスキーにより一意に識別されます。シーケンスキーは、pub.soap.wsrm:createSequence サービスの入力として、または Web サービスコネクタの入力パラメータとしても提供されます。

- **クライアントシーケンス ID**信頼性の高いメッセージ処理クライアントにより生成され、信頼性の高いメッセージ処理シーケンスに関連付けられる一意の識別子。クライアントが特定のシーケンスの一部としてメッセージを識別できるように、信頼性の高いメッセージ処理サーバは、SOAP 応答を信頼性の高いメッセージ処理クライアントに送信する際に、クライアントシーケンス ID を指定します。
- **サーバシーケンス ID**信頼性の高いメッセージ処理サーバにより生成され、信頼性の高いメッセージ処理シーケンスに関連付けられる一意の識別子。サーバが特定のシーケンスの一部としてメッセージを識別できるように、信頼性の高いメッセージ処理クライアントは、SOAP 要求を信頼性の高いメッセージ処理サーバに送信するときに、サーバシーケンス ID を指定します。メッセージシーケンスのサーバシーケンス ID は、pub.soap.wsrm:createSequence サービスの出力パラメータ、または Web サービスコネクタの reliableMessagingInfo/responseReliableMessagingProperties/serverSequenceId 出力パラメータとして返されます。
- **確認応答**信頼性の高いメッセージ処理宛先から信頼性の高いメッセージ処理ソースへの応答。メッセージが宛先に正常に受信されたことを示します。

Integration Server での信頼性の高いメッセージ処理の使用

信頼性の高いメッセージ処理を使用するように Integration Server を設定する場合は、以下の点に留意してください。

- Integration Server は、WS-ReliableMessaging バージョン 1.1 のみをサポートします。
- ネームスペースプリフィックス `wsrm` は、`http://docs.oasis-open.org/ws-rx/wsrm/200702` の URI を表します。
- ネームスペースプリフィックス `wsrmp` は、`http://docs.oasis-open.org/ws-rx/wsrmp/200702` の URI を表します。
- WS-ReliableMessaging を使用するには、Web サービス記述子に標準の WS ポリシーを付加します。このポリシーには、信頼性の高いメッセージ処理のアサーションが含まれます。独自ポリシーの定義の詳細については、『*Web Services Developer's Guide*』を参照してください。WS ポリシーを Web サービス記述子に付加する方法については、『*webMethods Service Development Help*』を参照してください。
- WS-ReliableMessaging を使用するには、Integration Server が提供する ReliableMessaging という名前の事前定義済みの WS ポリシーを付加します。このポリシーは次のディレクトリから入手できます。

  Integration Server_directory \instances*instance_name*\config\wss\policies

- Integration Server は、信頼性の高いメッセージ処理の一意のシーケンスキーを使用して、Web サービスプロバイダとコンシューマ間で確実に交換される一連のメッセージの進捗を追跡します。
- Integration Server は、信頼性の高いメッセージ処理に対して、順序を守ったデリバーの保証のみをサポートします。
- Web サービスの信頼性の高いメッセージ処理を設定するには、Integration Server Administrator を使用します。デフォルトでは、Integration Server は [**設定**] > [Web サービス] > [信頼性の高いメッセージ処理] > [設定の編集] ページで定義される信頼性の高いメッセージ処理設定を、すべての Web サービスプロバイダおよびコンシューマに適用します。特定の Web サービスプロバイダまたはコンシューマのサーバレベルの信頼性の高いメッセージ処理設定を変更するには、関連する Web サービスエンドポイントエイリアスの信頼性の高いメッセージ処理プロパティを定義します。
- Integration Server は、各種の pub.soap.wsrm 組み込みサービスを提供して、信頼性の高いメッセージ処理シーケンスを作成および管理します。これらの組み込みサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

信頼性の高いメッセージ処理データを対象とした永続記憶領域のサポート

Integration Server は、信頼性の高いメッセージ処理トランザクションデータのために、永続記憶領域の機能を提供します。永続記憶領域のサポートにより、信頼性の高いメッセージ処理ソースと信頼性の高いメッセージ処理宛先の間で交換されるメッセージは、システム障害や通信障害が発生しても失われません。メッセージが分散システムで交換される場合、通信リンクでメッセージを転送するとき、または、システムコンポーネントでメッセージを処理するときに、エラーが発生する可能性があります。このような状況でも、Integration Server では、メッセージが失われず、システム障害の後でもメッセージを最終的に回復できます。

Integration Server は、重要なルーティング情報やデリバー情報など、信頼性の高いメッセージ処理シーケンスに関連する情報を、永続記憶領域に保存する機能をサポートします。

Integration Server を設定して、信頼性の高いメッセージ処理に永続記憶領域の機能を提供する場合、以下の点に留意してください。

- Integration Server が永続記憶領域用に使用する必要がある、組み込み ISInternal データベースまたは外部 RDBMS のいずれかをポイントするように、ISInternal 機能エイリアス (**[設定] > [JDBC プール]** ページで指定) を設定する必要があります。
- 信頼性の高いメッセージ交換に使用する Integration Server がクラスタ環境にあり、同じデータベースに接続する場合、すべての Integration Server に対して、**[設定] > [Web サービス] > [信頼性の高いメッセージ処理] > [設定の編集]** ページで同じ永続設定を定義する必要があります。
- 信頼性の高いメッセージ処理のために Integration Server を設定して永続記憶領域の機能を提供する場合に限り、Integration Server はクラスタ環境で信頼性の高いメッセージ処理をサポートします。

- 認証の詳細が Integration Server の再起動後にも保持されるように、関連するコネクタの署名内ではなく、関連するコンシューマエンドポイントのエイリアス内で、コンシューマ Web サービス記述子に、認証の詳細を提供する必要があります。

Integration Server で信頼性の高いメッセージ処理を使用する場合の制限事項

- Integration Server は、WS-ReliableMessaging バージョン 1.1 のみをサポートします。
- Integration Server は、JMS を介した信頼性の高いメッセージ処理をサポートしません。
- WS セキュリティと WS-ReliableMessaging 標準の相互運用性の問題により、Integration Server は、付加された信頼性の高いメッセージ処理ポリシーに WS-ReliableMessaging と WS セキュリティの両方のポリシーアサーションが含まれる場合、信頼性の高いメッセージ処理をサポートしません。
- Integration Server は以下の WS-ReliableMessaging アサーションをサポートしません。
  - `wsrmp:SequenceSTR`
  - `wsrmp:SequenceTransportSecurity`

Integration Server での信頼性の高いメッセージ処理の設定

Web サービスの信頼性の高いメッセージ処理を設定するには、Integration Server Administrator を使用します。

信頼性の高いメッセージ処理プロパティを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [Web サービス] に移動します。
3. [**信頼性の高いメッセージ処理**] をクリックします。
4. [**信頼性の高いメッセージ処理の設定の編集**] をクリックします。
5. [**信頼性の高いメッセージ処理プロパティ**] で、以下の情報を指定します。

フィールド	指定内容
[再伝送間隔]	信頼性の高いメッセージ処理ソースが SOAP メッセージを再送信する前に信頼性の高いメッセージ処理宛先からの確認応答を待機する間隔 (ミリ秒単位)。デフォルトは 6000 ミリ秒です。
[確認応答間隔]	信頼性の高いメッセージ処理宛先がメッセージシーケンスの確認応答を送信する前に待機する間隔 (ミリ秒単位)。指定した確認応答間隔内に受信された同じシーケンスのメッセージは、1

フィールド	指定内容
	つのバッチで確認応答されます。確認応答間隔として指定された時間内に確認応答エンドポイントへその他のメッセージが送信されない場合、確認応答はスタンドアロンメッセージとして送信されます。 デフォルトは 3000 ミリ秒です。
[指数バックオフ]	指数バックオフアルゴリズムを使用して、確認応答のないメッセージの再送信間隔を調整するかどうかを指定します。再送信試行の間隔を調整することによって、再送信された大量のメッセージが信頼性の高いメッセージ処理宛先に集中しないようにします。 **選択項目** / **目的** **true** — 指定した再伝送間隔に基づいて、次の再送信までの間隔を指数的に延長します。たとえば、再伝送間隔を 2 秒と指定し、指数バックオフ値を true に設定すると、メッセージの確認応答がない状態が続いた場合、次の再送信までの間隔は 2、4、8、16、32 のようになります。これがデフォルトです。 **false** — すべての再送信に対して、[**再伝送間隔**] フィールドで指定されている同一の間隔を使用します。
[非活動状態タイムアウト間隔]	信頼性の高いメッセージ処理ソースが SOAP メッセージの再送信を停止する前に信頼性の高いメッセージ処理宛先からの確認応答を待機する時間 (ミリ秒単位)。単位を秒、分、時間または日で指定します。 指定した非活動状態タイムアウト内に信頼性の高いメッセージ処理ソースが確認応答を受信しない場合、シーケンスにはタイムアウトのマークが付けられます。タイムアウトしたシーケンスは使用できません。非活動状態タイムアウトの制限がないことを指定するには、[**非活動状態タイムアウト間隔**] を「-1」に設定します。 デフォルトは 60 秒です。
[シーケンス削除タイムアウト間隔]	シーケンスが終了した後に、メモリからシーケンスの状態を削除するまで信頼性の高いメッセージ処理ソースが待機する時間。単位を秒、分、時間または日で指定します。デフォルトは 600 秒です。
[配送順を保障]	シーケンス内のメッセージが、信頼性の高いメッセージ処理ソースによって送信されたのと同じ順序で信頼性の高いメッセージ処理宛先にデリバーされる必要があるかどうか。 **選択項目** / **目的**

フィールド	指定内容	
	[True]	シーケンス内のメッセージが送信されたのと同じ順序で宛先にデリバーされる必要があることを指定します。これがデフォルトです。
	[False]	送信されたのと同じ順序でのメッセージのデリバーを適用しません。
[最大再伝送回数]	信頼性の高いメッセージ処理宛先からの確認応答を受信しなかった場合に、信頼性の高いメッセージ処理ソースがメッセージを再送信する回数。再送信の試行回数に制限がないことを指定するには、[**最大再伝送回数**] を「-1」に設定します。デフォルトは 10 です。	
[記憶領域タイプ]	信頼性の高いメッセージ処理シーケンス情報を保存するために、Integration Server が永続モードと非永続モードのどちらを使用するかを指定します。	
	選択項目	**目的**
	[非永続]	信頼性の高いメッセージ処理シーケンス情報のために、非永続モードの記憶領域を使用します。[**非永続**] モードの記憶領域を使用する場合、Integration Server は信頼性の高いメッセージ処理データを保存するためにオンヒープメモリを使用します。Integration Server を再起動すると、信頼性の高いメッセージ処理情報はメモリから削除されます。これがデフォルトです。
	[データベース]	信頼性の高いメッセージ処理情報のために、永続モードの記憶領域を使用します。[**データベース**] モードの記憶領域を使用する場合、Integration Server は信頼性の高いメッセージ処理情報を保存するためにデータベースを使用します。重要なルーティング情報やデリバー情報など、信頼性の高いメッセージ処理シーケンスに関連するすべての情報が、Integration Server の再起動後も保持されます。
[ハウスキーピング間隔]	タイムアウトした、または終了したシーケンスをチェックするために、Integration Server がデータベースをスイープする間隔(秒)。 [**非活動状態タイムアウト間隔**] および [**シーケンス削除タイムアウト間隔**] の指定値に応じて、メッセージはタイムアウトしたり終了したりします。Integration Server は [**ハウスキーピング間隔**] に基づいて定期的にデータベースをスイープし、タイムアウトになったメッセージを識別してマークし、終了したメッセージを削除します。	

フィールド	指定内容
	デフォルトは 20 秒です。

6. [**変更内容の保存**] をクリックします。

信頼性の高いメッセージ処理のシーケンスレポート

シーケンスレポートには、Integration Server で処理中か処理が完了した信頼できるメッセージのシーケンスに関する情報が表示されます。このレポートには、クライアントおよびサーバのシーケンスが含まれています。つまり、信頼性の高いメッセージ処理クライアントとしての Integration Server が送信/受信した、および信頼性の高いメッセージ処理サーバとしての Integration Server が受信/送信した、信頼できるメッセージのシーケンスです。

クライアントおよびサーバのシーケンス

フィールド	説明
[ユーザシーケンスキー]	メッセージシーケンスを識別する一意のキー。[**ユーザシーケンスキー**] の値は、pub.soap.wsrm:createSequence サービスの sequenceKey 入力パラメータで指定された値、または Web サービスコネクタの reliableMessagingProperties/sequenceKey パラメータの値と同じです。
[クライアントシーケンス ID]	信頼性の高いメッセージ処理クライアントにより生成され、信頼性の高いメッセージ処理シーケンスに関連付けられる一意の識別子。
[サーバシーケンス ID]	信頼性の高いメッセージ処理サーバにより生成され、信頼性の高いメッセージ処理シーケンスに関連付けられる一意の識別子。この ID は、Integration Server により、pub.soap.wsrm:createSequence サービスの serverSequenceId 出力パラメータとして、または Web サービスコネクタの reliableMessagingInfo/responseReliableMessagingProperties/serverSequenceId 出力パラメータとして返されます。
[宛先アドレス]	信頼性の高いメッセージ処理シーケンスとの接続が確立される先のエンドポイントアドレス。
[状態]	メッセージシーケンスの状態。
[完了したメッセージの数]	デリバーされて確認応答メッセージを受信した、メッセージシーケンス内のメッセージの数。
[最後にアクティブになった日時]	シーケンス内の最後のメッセージが送信された時間を示すタイムスタンプ。
[確認応答要求]	信頼性の高いメッセージ処理サーバに確認応答要求を送信します。確認応答要求の送信の詳細については、428 ページの「確認応答要求の送信」を参照してください。

フィールド	説明
[閉じる]	シーケンスを閉じます。シーケンスを閉じる場合の詳細については、427 ページの「シーケンスを閉じる」を参照してください。
[強制終了]	シーケンスを終了します。シーケンスを終了する場合の詳細については、428 ページの「シーケンスの終了」を参照してください。

信頼性の高いメッセージ処理のシーケンスレポートの表示

メッセージシーケンスの詳細は、Integration Server Administrator の [信頼性の高いメッセージ処理] ページに表示されます。

信頼性の高いメッセージ処理のシーケンスレポートを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. **[設定]** > [Web サービス] に移動します。
3. **[信頼性の高いメッセージ処理]** をクリックします。

   Integration Server では、**[クライアントシーケンス]** および **[サーバシーケンス]** の下にシーケンスレポートが表示されます。

シーケンスを閉じる

信頼性の高いメッセージ処理サーバにより既に受け入れられたメッセージおよび処理中のメッセージ以外に、指定されたシーケンスを使用して信頼性の高いメッセージ処理クライアントが新しいメッセージを送信する必要がなくなった場合、メッセージシーケンスを閉じることができます。

信頼性の高いメッセージ処理シーケンスを閉じると、信頼性の高いメッセージ処理サーバは、同じシーケンスのいかなる新しいメッセージも受け入れません。ただし、信頼性の高いメッセージ処理サーバは、閉じたシーケンスの追跡と閉じたシーケンス内のメッセージに対する確認応答要求への応答は続行します。

メッセージシーケンスを閉じることができるのは、信頼性の高いメッセージ処理クライアントだけです。信頼性の高いメッセージ処理シーケンスを閉じると、クライアントのみがシーケンスを終了できます。指定された非活動状態タイムアウト期間内に、閉じたシーケンスに関連するアクティビティがない場合、信頼性の高いメッセージ処理クライアントはこのシーケンスをタイムアウトとしてマークします。

シーケンスを閉じるには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. **[設定]** > [Web サービス] に移動します。

3. [**信頼性の高いメッセージ処理**] をクリックします。

4. [**クライアントシーケンス**] の下で、閉じるシーケンスを見つけて、[**閉じる**] リンクをクリックします。

シーケンスの終了

シーケンス内のすべてのメッセージの確認応答を待機せずに、信頼性の高いメッセージ処理シーケンスを終了できます。メッセージシーケンスを終了することができるのは、信頼性の高いメッセージ処理クライアントだけです。信頼性の高いメッセージ処理クライアントからのシーケンス終了要求は、シーケンスが完了し、クライアントからはシーケンスに関連するメッセージがこれ以上送信されないことを示しています。ただし、メッセージシーケンスのシーケンスキーは、[**シーケンス削除タイムアウト間隔**] 期間の経過後のみ再利用できます。

シーケンスを終了するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [Web サービス] に移動します。

3. [**信頼性の高いメッセージ処理**] をクリックします。

4. [**クライアントシーケンス**] の下で、終了するシーケンスを見つけて、[**強制終了**] リンクをクリックします。

確認応答要求の送信

信頼性の高いメッセージ処理クライアントが、信頼性の高いメッセージ処理サーバに特定のメッセージシーケンスの確認応答を要求するように指示できます。

確認応答要求を送信するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [Web サービス] に移動します。

3. [**信頼性の高いメッセージ処理**] をクリックします。

4. [**クライアントシーケンス**] の下で、信頼性の高いメッセージ処理サーバに確認応答要求を送信する対象のシーケンスを見つけます。[**確認応答要求**] の下で、[**送信**] リンクをクリックします。

20 JWT を使用するように Integration Server を設定する

- JWT の概要 430
- Integration Server での JWT の使用 431
- Integration Server における JWT のサポート 432
- JWT を使用するように Integration Server を設定する 433

JWT の概要

JSON Web Token (JWT) は、2 者間でクレームを交換する安全な手段を提供する JSON ベースの標準 (RFC 7519) です。JWT はクレームを簡潔な形式で表します。これは、HTTP 認証ヘッダーなど、スペースが限られた環境を対象としています。JWT では、クレームを JSON オブジェクトとしてエンコードします。JSON オブジェクトは JWT のペイロードとして使用されます。

ユーザ状態はサーバメモリに保存されないため、JWT 認証はステートレスです。JWT は独立しています。つまり、ユーザに関して必要な情報をすべて含んでいます。

JWT の構造

JWT の構造は次の要素から構成されます。

- **ヘッダー**。トークンタイプおよび使用されるハッシュアルゴリズムを識別します。この場合、トークンタイプは JWT です。
- **ペイロード**。JWT クレームを含みます。クレームは、ユーザがサーバに送信するエンティティ (ユーザ) および追加メタデータに関するステートメントです。次に、クレームのタイプを示します。
  - **登録済みクレーム**。必須ではないが推奨される一連の事前定義済みクレームです。
  - **パブリッククレーム**。一連の定義済みクレームです。
  - **プライベートクレーム**。使用について同意している当事者間で情報交換のために作成された一連のカスタムクレームです。
- **署名**。JWT ヘッダーとペイロードの整合性を確認します。

したがって、JWT は `header.payload.signature` のようになります。

登録済みクレーム

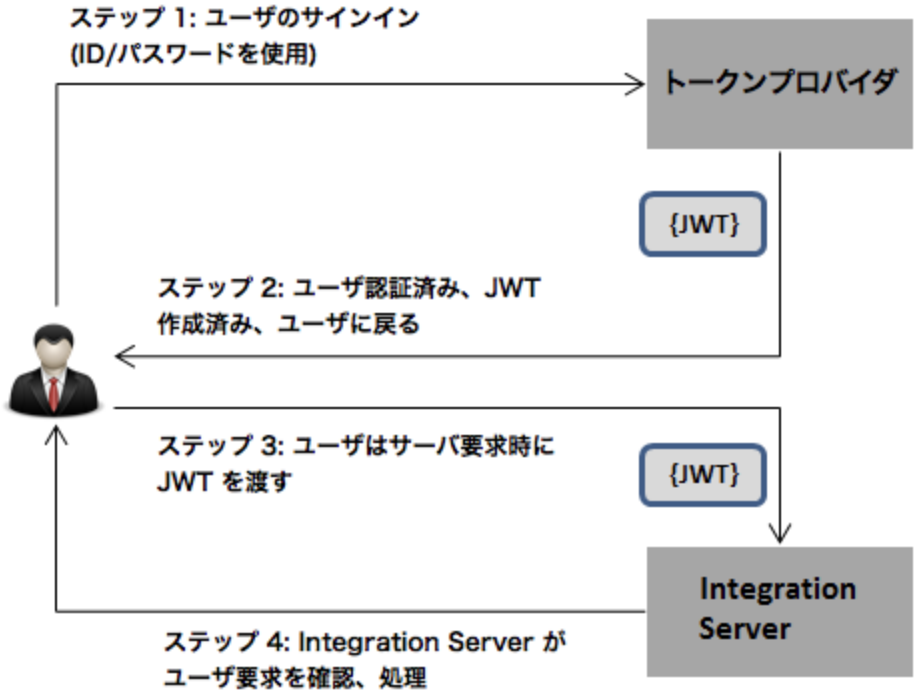
登録済みクレームは IANA JSON Web トークンクレームレジストリに登録されます。次に、一般に使用される登録済みクレームを示します。

クレーム名	説明
`iss`	Issuer (発行元) JWT を発行したプリンシパルを示します。
`sub`	Subject (サブジェクト) JWT のサブジェクトであるプリンシパルを示します。
`aud`	Audience (オーディエンス) JWT が対象とする受信者を示します。JWT を処理する予定の各プリンシパルはオーディエンスクレームの値で自身を識別する必要があります。

クレーム名	説明
exp	Expiration Time (有効期限) JWT の有効期限を示します。これ以降に JWT を処理してはいけません。
nbf	Not Before JWT の処理が可能になる時刻を示します。

Integration Server での JWT の使用

Integration Server を使用して JWT を有効にし、設定できます。下図は、JWT 認証に関する手順を示します。



ステップ 1	ユーザサインイン (トークンプロバイダへ)。 Integration Server にある保護されたデータまたはサービスにアクセスするには、クライアントが自分のクレデンシャルを使用してサードパーティのトークンプロバイダに JWT ベアラートークンの要求を送信します。
ステップ 2	JWT を取得します。 認証後、トークンプロバイダは JWT を作成し、クライアントに返します。
ステップ 3	ユーザが JWT を使用します。 クライアントが Integration Server のリソースまたはサービスにアクセスする要求を送信するごとに、クライアントは要求とともこのトークンのコピー

を Integration Server に送信します。クライアントは、自身を認証するベアラースキームを使用して認証ヘッダーで JWT を送信します。

ステップ 4 Integration Server によって、JWT の妥当性検査が行われます。

Integration Server Administratorで定義されたマッピングを認証する発行元に基づいて、Integration Server は承認エイリアスを識別し、対応する公開鍵を使用して署名を検証します。署名の検証が失敗すると、要求は拒否されます。

注:
Integration Server では未署名の JWT (署名のない JWT) をサポートしていません。

署名の検証後、Integration Server は JWT ペイロードに含まれているクレームを検証します。

クレーム名	Integration Server が検証する内容
`iss`	発行元 ID が、Integration Server に定義されている信用のある発行元リストに一致する。
`sub`	ユーザが Integration Server にマップされている。
`aud`	オーディエンスの値が、Integration Server のグローバルクレーム設定に定義されたオーディエンスに一致する。これはリストまたは単一の値です。リストである場合、Integration Server の定義済みオーディエンスはこのリストの値の 1 つです。
`exp`	有効期限が現在の時刻より後になっている。トークンを拒否する前のスキューマッピングにおける Integration Server 要素です。
`nbf`	指定時間後が現在の時刻より前になっている。トークンを拒否する前のスキューマッピングにおける Integration Server 要素です。

上記のクレームのいずれかの検証が失敗した場合、Integration Server では要求を拒否します。

ステップ 5 Integration Server がユーザに応答します。

妥当性検査の後、Integration Server は要求されたデータまたは該当するエラーメッセージでクライアントに応答します。

Integration Server における JWT のサポート

現在、Integration Server では次の JWT 機能のみサポートしています。

- 登録済みクレーム

- RS256、RS384、RS512 などの RSA アルゴリズム

JWT を使用するように Integration Server を設定する

JWT を使用するように Integration Server を設定するには、次のステージを実行します。

ステージ 1 信用のある発行元を追加します。

Integration Server では、受信 JWT の発行元 ID (`iss` クレーム) を検証し、発行元 ID が Integration Server で保持されている信用のある発行元リストに一致するかどうかを確認します。信用のある発行元の追加と編集の詳細については、434 ページの「信用のある発行元」を参照してください。

ステージ 2 発行元と認証のマッピングを定義します。

このステージでは、信用のある発行元と認証のマッピングを定義します。このステージで定義された発行元と認証のマッピングに基づいて、Integration Server は承認エイリアスを識別し、その認証からの公開鍵を使用して受信 JWT の署名を検証します。発行元と認証のマッピングの定義および既存のマッピングの削除については、435 ページの「発行元と認証のマッピング」を参照してください。

ステージ 3 グローバルクレーム設定を編集します。

- [**オーディエンス**]: Integration Server では、受信 JWT に定義されているオーディエンスを検証し、オーディエンスがグルーバルクレーム設定に定義されているリストに一致するかどうかを確認します。オーディエンスの値はリストまたは単一の値です。リストである場合、Integration Server の定義済みオーディエンスはこのリストの値の 1 つです。グローバルクレーム設定を編集する方法については、437 ページの「グローバルクレーム設定の編集」を参照してください。
- [**最大グローバルスキュー**]: JWT 発行元サーバクロックと Integration Server クロック間のずれの許容可能な上限を定義できます。Integration Server は受信 JWT の検証時にこの値を使用します。JWT クロックスキュー設定の詳細については、436 ページの「スキューマッピング」を参照してください。

Integration Server での JWT の拡張サポート

既存の JWT 機能をサポートするのに加え、Integration Server は JWT の生成と妥当性検査およびクレームの抽出をサポートします。

次の表では、JWT パブリックサービスを示します。

サービス	説明
pub.jwt:extractClaimsFromJWT	JWT から抽出する特定のクレーム詳細を返します。

サービス	説明
pub:jwt:generateSignedJWT	署名付き JWT を生成します。
pub:jwt:verifyJWT	JWT の署名、トークンの有効期限、および指定時間後を検証します。

JWT のパブリックサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

信用のある発行元

信用のある発行元は、公開認証が Integration Server に保存されている発行元です。

Integration Server では信用のある発行元のリストを保持します。Integration Server Administrator を使用して、信用のある発行元を追加できます。既存の信用のある発行元を編集または削除することもできます。

信用のある発行元の追加

信用のある発行元を追加するには、以下の手順に従います。

信用のある発行元を追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [**JWT**] に移動します。
3. [**信用のある発行元**] をクリックします。
4. [**発行元の追加**] をクリックします。
5. [発行元の追加] ページの [**名前**] フィールドに発行元の名前を入力します。

   注:
   発行元の名前は大文字と小文字を区別します。

6. [**説明**] フィールドに、発行元の説明を入力します。
7. [**変更内容の保存**] をクリックします。

   一度信用のある発行元を追加すると、その発行元の説明のみ編集できます。

信用のある発行元の削除

信用のある発行元を削除するには、以下の手順に従います。

注:
マッピング (発行元と認証のマッピング) が存在する発行元を削除することはできません。

信用のある発行元を削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [**JWT**] に移動します。
3. [**信用のある発行元**] をクリックし、削除する発行元をクリックします。
4. [信用のある発行元] で削除する発行元の [削除] 列の ✕ をクリックします。
5. [**はい**] をクリックして確認します。

   Integration Server で信用のある発行元が削除されます。

発行元と認証のマッピング

Integration Server Administrator を使用して、Integration Server にリストされている信用のある発行元とトラストストアの認証間のマッピングを作成できます。また、既存のマッピングを削除することもできます。

発行元と認証のマッピングの作成

発行元と認証のマッピングによって、信用のある発行元とトラストストアの認証間の関連付けが確立されます。Integration Server では、受信 JWT の署名の検証中に発行元と認証のマッピングを使用します。

発行元と認証のマッピングを作成する前に、必要な信用のある発行元、トラストストアエイリアス、および認証エイリアスが Integration Server に存在することを確認します。

発行元と認証のマッピングを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [**JWT**] に移動します。
3. [**発行元の設定**] > [**発行元と認証のマッピングの追加**] をクリックします。
4. [**発行元と認証のマッピングの追加**] ページで、次の情報を指定します。

フィールド	選択項目
[発行元]	認証にマップする信用のある発行元の名前。

フィールド	選択項目
[トラストストアエイリアス]	選択された発行元と関連付けられた署名機関の認証を含むトラストストアのトラストストアエイリアス。
[認証エイリアス]	トラストストアエイリアスに関連付けられた認証の認証エイリアス。

5. [**変更内容の保存**] をクリックします。

発行元と認証のマッピングの削除

無効または期限切れの認証、またはマッピングに関連して不要になった認証の一部を削除する必要がある場合、発行元と認証のマッピングを削除できます。

発行元と認証のマッピングを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [**JWT**] に移動します。
3. [**発行元の設定**] をクリックします。
4. 削除するマッピングの [**発行元と認証のマッピング**] テーブルで、[**削除**] 列の ✕ をクリックします。
5. [**はい**] をクリックして確認します。

   Integration Server で発行元と認証のマッピングが削除されます。
6. [削除] 列の ✕ アイコンをクリックします。

スキューマッピング

異なるサーバのクロックは数秒異なることがあります。JWT がタイムスタンプを使用するとき、JWT 発行元サーバクロックと Integration Server クロック間にずれがあると、有効な JWT トークンが拒否されることがあります。スキューマッピングを使用すると、そのようなクロックの違いに指定した上限まで対応できます。Integration Server は受信 JWT の検証時にスキューマッピング設定を使用します。

スキューマッピングはグローバルレベル (437 ページの「グローバルクレーム設定の編集」 を参照) または発行元レベル (436 ページの「発行元スキューマッピングの作成」 を参照) で定義できます。発行元スキューマッピング値とグローバルスキューマッピング値の両方が利用できる場合、発行元スキューマッピング値が優先されます。

発行元スキューマッピングの作成

発行元スキューマッピングを作成する前に、必要な信用のある発行元が Integration Server に存在することを確認します。

発行元スキューマッピングを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [**JWT**] に移動します。
3. [**発行元の設定**] > [**発行元スキューマッピングの追加と更新**] をクリックします。
4. [**発行元スキューマッピングの追加と更新**] ページで、次の情報を指定します。

フィールド	選択項目
[発行元]	信用のある発行元の名前。
[スキュー (秒)]	JWT 発行元と Integration Server のクロックが異なっても、JWT の有効期限 (exp) と指定時間後 (nbf) の検証が成功できるようにする最大秒数。この値は [**最大グローバルスキュー**] の値よりも優先されます。

5. [**変更内容の保存**] をクリックします。

グローバルクレーム設定の編集

グローバルクレーム設定には、Integration Server で受信 JWT のクレームを検証するときに検索される JWT のプロパティが保持されます。

グローバルクレーム設定を定義するには、以下の手順に従います。

グローバルクレーム設定を編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [**JWT**] に移動します。
3. [**グローバルクレーム設定の編集**] をクリックし、以下の詳細を指定します。
   - [**オーディエンス**]: 有効な文字列または URI。
   - [**最大グローバルスキュー (秒)**]: JWT 発行元と Integration Server のクロックが異なっても、JWT の有効期限 (exp) と指定時間後 (nbf) の検証が成功できるようにする最大秒数を指定します。正の整数またはゼロを指定します。

     注:
     発行元の発行元スキューマッピングが利用できない場合、この値が使用されます。

4. [**変更内容の保存**] をクリックします。

21 ケルベロスを使用するようにIntegration Serverを設定する

- Integration Server のケルベロス使用方法の概要 440
- ケルベロスについて 440
- ケルベロス委任認証 441
- ケルベロス設定の前提条件 443
- Integration Server でケルベロス認証を使用するときの制限事項 443
- ケルベロスを使用するように Integration Server を設定する 444
- ケルベロスの JAAS コンテキスト 447
- ケルベロス設定のトラブルシューティング 447

Integration Server のケルベロス使用方法の概要

ケルベロスは、クライアントの識別情報の妥当性を検査するために、対称暗号化および信頼できるサードパーティシステムを使用する認証プロトコルの一種です。ケルベロスプロトコルは、ホスト間の通信が傍受される可能性のあるオープンでセキュリティ保護されていないネットワーク上で認証を提供します。

Integration Server を使用して、サービス要求のためのケルベロス認証を有効化し設定することができます。Integration Server は、次のタイプの要求でケルベロス認証の使用をサポートしています。

- 転送レベルの受信/送信 HTTP 要求
- 受信/送信 HTTPS Web サービス要求

注:
現在、Integration Server は transportType *HTTPS* の送信 HTTPS Web サービス要求のケルベロス認証のみサポートしています。

注:
Integration Server は、JDBC 接続プールによるデータベースへの送信接続にケルベロス認証を使用することをサポートしています。詳細については、165 ページの「データベースへの送信接続にケルベロス認証を使用する」を参照してください。

ケルベロスについて

ケルベロス 認証システムは、以下の要素で構成されます。

- ケルベロスクライアント。ケルベロスサービスにアクセスして使用するために必要です。
- 信頼できるサードパーティシステム。具体的には、KDC (Key Distribution Center : キー配信センター)。
- サーバ。ケルベロス認証を使用してアクセス可能なサービスをホストします。

ケルベロス認証は、次のフェーズで構成されます。

1. **認証フェーズ**クライアントは、自分自身を認証サービスに対して認証し、長期間のチケット付与チケット (TGT) を要求します。
2. **サービス認証フェーズ**クライアントは、TGT を使用して、呼び出しを希望する特定サービスのチケットを要求します。
3. **サービス呼び出しフェーズ**クライアントは、要求を送信して、サービス認証フェーズで取得されたサービスチケットを含むターゲットサービスを呼び出します。要求されたサービスをホストするサーバがサービスチケットを認証する場合、サーバは要求されたサービスを呼び出します。

ケルベロス用語

Integration Server で ケルベロス認証の使用を設定する前に、以下の用語を理解しておくと便利です。

- **キー配布センター (KDC)**信頼できるサードパーティシステム。認証およびチケット付与サービスを提供し、プリンシパルデータベースをホストします。つまり KDC は、認証サーバ、チケット付与サーバ、ならびにプリンシパルとその鍵が格納されたデータベースで構成されます。
- **領域**KDC およびセカンダリ KDC (存在する場合) による管理対象であるすべてのコンピュータによって領域が構成されます。つまり、領域には、同じケルベロスデータベースを共有するすべてのノードが含まれます。
- **プリンシパル**ケルベロスシステムによって認識されるサービスまたはユーザ。各ケルベロスプリンシパルは、プリンシパル名で識別されます。プリンシパル名は、*principal-name.instance-name@realm-name* という形式で、サービスまたはユーザの名前、インスタンス名、領域名という 3 つの部分で構成されます。
- **サービスプリンシパル名**プリンシパルデータベースに登録された、サービスのプリンシパル名。
- **キータブファイル**一連のプリンシパルとそのパスワードで構成されるファイル。さまざまな暗号化アルゴリズムを使用して暗号化されたプリンシパルパスワードを格納できます。
- **ケルベロス設定ファイル**ケルベロス領域、KDC の場所、現在の領域のデフォルト設定、暗号化アルゴリズムなど、設定情報が格納されたファイル。一般にこのファイルの名前は「krb5.conf」です。
- **サブジェクト**JAAS ログインコンテキストで認証されるユーザまたはサービス。

ケルベロス委任認証

ケルベロス委任を使用すると、Integration Server はクライアントのクレデンシャルを再使用して、同じサーバまたは異なるサーバにホストされている別のサービスを呼び出すことができます。この場合、プリンシパル (委任されたユーザ) は別のプリンシパル (元の要求者) の代わりにサービスを呼び出すことができます。

ケルベロス委任認証は、次のフェーズで構成されます。

1. **認証フェーズ**クライアントは、自分自身を認証サービスに対して認証し、転送可能な TGT (委任可能なトークン) を要求します。
2. **サービス認証フェーズ**
   a. クライアントは転送可能な TGT を使用して、媒介のサービスチケットを要求します。
   b. 媒介は転送可能な TGT を使用して、元の要求者の代わりにサービスのサービスチケットを要求します。

注:

チケットは任意の数の媒介に転送できます。媒介の数に関係なく、サービス要求は元の要求者の代わりに呼び出されます。

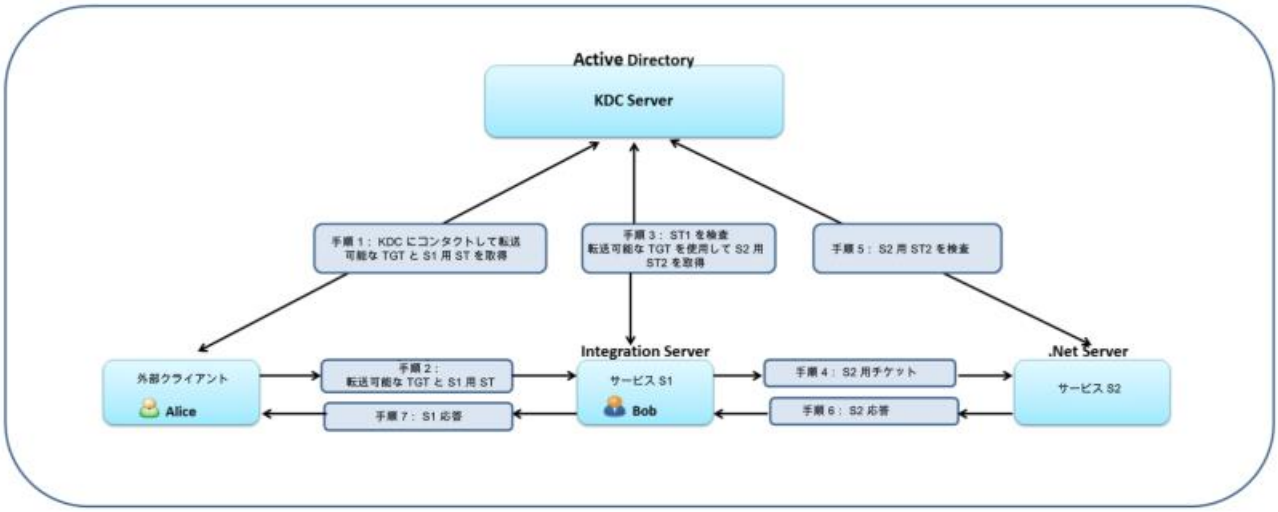
3. **サービス呼び出しフェーズ**媒介は、要求を送信して、サービス認証フェーズ (ステップ 2.b) で取得されたサービスチケットを含むターゲットサービスを呼び出します。

使用事例

次の使用事例では、ケルベロス委任認証が利用される手順を説明します。

この使用事例では、以下の点を考慮します。

- 外部クライアント、媒介 (Integration Server)、および宛先サーバ (.Net Server) は同じ KDC を共有します。
- 外部クライアントはユーザアカウントAlice を使用して、媒介 (Integration Server) のサービス S1 にアクセスします。
- 媒介はユーザアカウント Bob を使用して、.Net Server にホストされたエンドポイントサービス S2 を呼び出します。
- Alice は S1 を呼び出します。



設定:

- S1 で、入力パラメータ `delegation` は http 送信呼び出し (pub.client:http) で *kerberos* に設定されています。

手順:

1. 外部クライアントは KDC にコンタクトし、転送可能な TGT および Alice のサービスチケット (ST1) を要求します。
2. 外部クライアントは転送可能な TGT と ST1 を使用して S1 を呼び出します。

3. Integration Server はトークンを受け取り、次に転送可能な TGT の妥当性検査を行い、その TGT を抽出します。Integration Server は Bob としてログインし、Alice の転送可能な TGT を使用して S2 のサービスチケット (ST2) を要求します。
4. Integration Server は ST2 を使用してターゲットサービス S2 を呼び出します。
5. .Net Server は KDC にコンタクトし、ST2 の妥当性検査を行います。サービスが呼び出されます。
6. Integration Server は S2 に対する応答を受け取ります。
7. Integration Server は応答 (成功または失敗) を外部クライアントに転送します。

ケルベロス設定の前提条件

ケルベロス認証を使用するように Integration Server を設定するには、以下の点を確認する必要があります。

- 動作するキー配布センター (KDC) が設定されている
- ケルベロスチケットを持つ受信要求の認証のために、KDC が LDAP ディレクトリとして設定されている
- Integration Server がクライアントによってサブミットされたケルベロスチケットの一部であるユーザを識別し認証できるように、KDC が Central Users または LDAP のユーザディレクトリとして設定されているケルベロス認証の詳細については、552 ページの「ケルベロス認証」を参照してください。
- ケルベロスクライアントが KDC のプリンシパルデータベースに登録されている
- アクセスするサービスが KDC に登録されている
- 有効なケルベロス設定ファイルが使用可能である

Integration Server でケルベロス認証を使用するときの制限事項

- Integration Server におけるケルベロス認証では、現在のところ `com.sun.security.auth.module.Krb5LoginModule` で提供されているケルベロスログインモジュールのみをサポートします。
- Integration Server は、`javax.security.auth.useSubjectCredsOnly` などのシステムプロパティがクライアントシステム上に既に存在する場合は、そのプロパティを上書きすることがあります。
- キータブ設定は、JDK 1.7_45 バージョン以上で認定されます。
- Web サービス要求に対するケルベロス委任認証は現在サポートされていません。

ケルベロスを使用するように Integration Server を設定する

Integration Server Administrator を使用して、受信/送信サービス要求のためにケルベロス認証を有効化して使用するように Integration Server を設定します。Integration Server は、この情報を ケルベロスログインモジュールに渡します。

Integration Server のケルベロスを設定する場合は、以下の情報に留意してください。

- [**領域**] および [**キー配布センターホスト**] に指定した値は、[**ケルベロス設定ファイル**] に指定されているキー配布センター (KDC) 設定ファイルのデフォルト KDC および領域セットを上書きします。ケルベロス設定ファイルのデフォルト KDC および領域セットを上書きしたくない場合は、[**領域**] および [**キー配布センターホスト**] の値を指定しないでください。
- [**キー配布センターホスト**] を指定する場合は、[**領域**] を指定する必要があります。その逆も同様です。
- デフォルトでは、ケルベロス認証を必要とする送信要求に対して Integration Server は JGSS ケルベロス OID を使用して Java ケルベロスチケットを生成します。Integration Server が、ケルベロス認証を使用する送信要求に対して SPNEGO ベースのケルベロスチケットを生成する必要がある場合、サーバ設定パラメータ watt.security.kerberos.client.useSPNEGO を「true」に設定します。この設定によって、Integration Server は、SPNEGO OID (オブジェクト識別子) を使用して、ケルベロス認証を必要とするすべての送信要求に対して SPNEGO トークンを生成します。

ケルベロス認証を設定するには

1. Integration Server Administrator を開きます。
2. [**セキュリティ**] > [ケルベロス] に移動します。
3. [**ケルベロス設定の編集**] をクリックします。
4. **[セキュリティ] > [ケルベロス] > [編集]** ページの [ケルベロス設定] で、次の情報を入力します。

フィールド	指定する値
[領域]	ケルベロスサーバのドメイン名。すべて大文字で指定します。KDC およびセカンダリ KDC (存在する場合) による管理対象であるすべてのコンピュータによって領域が構成されます。 例: `KERBEROS.RNDLAB.LOC` このフィールドはオプションです。 **注:** [**領域**] で指定される値は、[**ケルベロス設定ファイル**] で指定される KDC 設定ファイルの領域セットを上書きします。

フィールド	指定する値
[キー配布センターホスト]	KDC が配置されているマシンのホスト名。 例: `lab.kerberos.rndlab.loc` このフィールドはオプションです。 **注:** **[キー配布センターホスト]** に指定した値は、**[ケルベロス設定ファイル]** に指定されている KDC 設定ファイルのデフォルトキー配布センターセットを上書きします。
[ケルベロス設定ファイル]	ケルベロス設定情報が格納されたケルベロス設定ファイルの場所。格納される情報には、KDC の場所、領域およびケルベロスアプリケーションのデフォルト設定、ホスト名とケルベロス領域のマッピングなどがあります。
[サブジェクトクレデンシャルのみ使用]	JAAS 認証モジュールによって設定された既存のサブジェクトから必要なクレデンシャルを取得するために、Integration Server でケルベロス V5 GSS (Generic Security Services) メカニズムが必要であるかどうかを指定します。ここでは、「サブジェクト」は JAAS ログインコンテキストで認証されるユーザまたはサービスを表します。 **[サブジェクトクレデンシャルのみ使用]** チェックボックスをオンにすると、Integration Server は既存サブジェクトからクレデンシャルを取得するために GSS メカニズムを必要とします。Integration Server はサブジェクトに格納されている TGT (Ticket Granting Ticket : チケット付与チケット) を使用して、GSS セキュリティコンテキストを確立します。サービスチケットもサブジェクトに格納されます。**[サブジェクトクレデンシャルのみ使用]** チェックボックスをオンにすると、Integration Server が実行されている JVM は、JAAS 認証モジュールのサブジェクト内にあるクレデンシャルのみ使用できます。JVM は、クレデンシャルの取得に別の基盤メカニズムは使用できません。 [サブジェクトクレデンシャルのみ使用] チェックボックスをオフにすると、Integration Server は既存サブジェクトからクレデンシャルを取得するのに GSS メカニズムを必要としません。その代わり、Integration Server が実行されている JVM はサブジェクトのクレデンシャルを取得するのに、ディスク上の保護ファイルからの読み取りなど、選択した別の基盤メカニズムを使用できます。JVM はまず、JAAS 認証モジュールのサブジェクトをチェックします。JVM は、JAAS サブジェクトでクレデンシャルを見つけられない場合、レデンシャルを取得するために別のクレデンシャルメカニズムを使用します。 サービス要求にケルベロス認証を使用したい場合は、**[サブジェクトクレデンシャルのみ使用]** をオンにする必要があります。

5. [**変更内容の保存**] をクリックします。

プリンシパル名およびパスワードの優先順位

ケルベロスログインモジュールでは、プリンシパル名とパスワードを使用してプリンシパルをキー配布センター (KDC) に対して認証します。ただし、プリンシパル名とパスワードは、他の場所で指定できます。

- プリンシパル名は JAAS ログイン設定ファイルである is_jaas.cnf ファイル、プリンシパルパスワードはキータブファイルで指定できます。特定のプリンシパル名およびキータブファイルで指定された対応するパスワードを使用するには、is_jaas.cnf ファイルで `principal=client_principal_name` および `useKeyTab=true` を指定する必要があります。

  注:
  プリンシパルを指定するためにこのモードを使用する場合は、ケルベロスログインモジュールの使用はそのプリンシパルに制限されます。

- 受信サービス要求では、ポート設定にもプリンシパル名とパスワードを指定できます。
- 送信サービス要求では、*auth\kerberos* ドキュメントの *clientPrincipal* および *clientPassword* フィールドの pub.client:http サービスでプリンシパル名とパスワードを指定する必要があります。
- 受信/送信 Web サービス要求では、Web サービスエンドポイントエイリアスにもプリンシパル名とパスワードを指定できます。
- 送信 Web サービス要求では、*auth\message\kerberosSettings* ドキュメントの *clientPrincipal* と *clientPassword* を使用して、実行時に Web サービスコネクタでもプリンシパル名とパスワードを指定できます。

サービス要求では、Integration Server は、使用するプリンシパル名およびパスワードを決定するときに、以下の順番の優先権が使用されます。

1. is_jaas.cnf ファイルのプリンシパル名と、キータブファイルで指定されている対応するパスワード。
2. 受信サービス要求では、ポート設定で指定されたプリンシパル名とパスワード (存在する場合)。

   送信サービス要求では、pub.client:http サービスで指定されたプリンシパル名とパスワード (存在する場合)。

Web サービス要求では、Integration Server は、使用するプリンシパル名およびパスワードを決定するときに、以下の順番の優先権が使用されます。

1. is_jaas.cnf ファイルのプリンシパル名と、キータブファイルで指定されている対応するパスワード。
2. 送信 Web サービス要求では、実行時に Web サービスコネクタで指定されたプリンシパル名とパスワード (存在する場合)。

3. 受信/送信 Web サービス要求では、実行時にランタイムに Web サービスエンドポイントエイリアスで指定されたプリンシパル名とパスワード (存在する場合)。

ケルベロスの JAAS コンテキスト

Integration Server で配布される is_jaas.cnf ファイルは、 *Integration Server_directory* \instances*instance_name*\config フォルダにあり、次の JAAS コンテキストを含んでいます。

- 受信要求で使用する IS_KERBEROS_INBOUND
- 送信要求で使用する IS_KERBEROS_OUTBOUND

is_jaas.cnf で指定されている JAAS コンテキストを使用するか、独自の値を追加できます。

重要:
ただし、デフォルト JAAS ログインコンテキスト (IS_KERBEROS_INBOUND、IS_KERBEROS_OUTBOUND、IS_Transport、PlatformManagement、WSS_Message_IS、WSS_Transport_IS、および WSS_Transport_ProxyService) は JAAS 設定ファイルから削除しないでください。

ケルベロス設定のトラブルシューティング

Web サービスコネクタを実行すると、「org.apache.axis2.AxisFault: Error in building kerberos token」というエラーを受信します。
このエラーの考えられる理由は以下のとおりです。

- JAAS コンテキストの形式が正しくない。
- 指定されたプリンシパル名に対して設定されたケルベロスログインモジュールがない。
- JAAS コンテキストで指定されたキータブファイルの絶対パスが正しくない。
- キータブファイル内の暗号化アルゴリズムが KDC で使用される暗号化アルゴリズムと異なる。
- 使用している JDK バージョンが JDK 1.7_45 未満である。キータブ設定は、JDK 1.7_45 バージョン以上で認定されます。

Web サービスコネクタを実行すると、「Pre-authentication information was invalid (24) KrbException: Identifier doesn't match expected value (906)」というエラーを受信します。
このエラーの考えられる原因は以下のとおりです。

- トラストストアまたはキータブファイルが破損している。このような場合、ファイルを再作成して、頻繁にアクセスされない場所に配置することを Software AG ではお勧めします。
- システム時刻と、KDC をホストしているシステムの時刻が一致しない。これら 2 つのシステムのクロックは、デフォルトのクロックスキューである 300 秒を超えて非同期状態になることはできません。

委任を *kerberos* に設定しケルベロス委任を要求すると、「ケルベロス委任可能クレデンシャルは存在しません。」というエラーを受信します。

このエラーの考えられる理由は、pub.client:http サービスの *requestDelegatableToken* パラメータが *true* に設定されていないことです。

22 HTTP URL エイリアスの設定

- 概要 450
- HTTP URL エイリアスの作成 451
- URL エイリアスの部分一致の有効化 457
- HTTP URL エイリアスの表示 457
- URL エイリアスの編集 458
- URL エイリアスの削除 459
- URL エイリアスの移植性と競合の可能性 459

概要

HTTP URL エイリアスとは、Integration Server 上のリソースへの URL の一部を置き換えるために作成するエイリアスです。URL エイリアスは通常、リソースの名前と場所を識別する URL のパスの部分を置き換えます。例えば、サービスの URL エイリアスを作成するときは、エイリアスは呼び出しディレクティブと Integration Server のサービスの名前と場所を置き換えます。サービスに対するクライアント要求は、呼び出しディレクティブとサービス情報の代わりにエイリアスを含みます。Integration Server が要求を受信すると、Integration Server はエイリアスに関連付けられたサービスを呼び出します。

URL エイリアスは、サービス、HTML ページ、DSP (Dynamic Server Pages: ダイナミックサーバページ)、画像ファイル、Web サービスなどの項目と関連付けられます。URL エイリアスを単一リソースに関連付けるのに加え、要求の受信ポートに基づいて 1 つの URL エイリアスで異なるリソースを識別することも可能です。これにより、HTTP 要求の受信ポートに基づいて異なる宛先を解決する 1 つの URL エイリアスの定義も可能になります。さらに、エイリアスとして「空のパス」を定義し、受信ポートのデフォルトの宛先を定義することも可能です。

URL エイリアスには以下のような利点があります。

- URL エイリアスは、完全な URL パス名よりも入力が簡単です。
- リソースに URL エイリアスがあると、Integration Server 上でリソースの移動が簡単になります。管理者は新しい場所を指し示すようにエイリアス情報を更新すればよく、変更についてユーザに通知する必要はありません。
- ユーザにはディレクトリもサービスもファイル名も表示されないため、URL エイリアスは URL パス名よりも安全です。

注:
URL エイリアスは HTTP ポートまたは HTTPS ポートでのみ使用できます。

URL エイリアスは、Designer または Integration Server Administrator で作成できます。

以下の操作を行う場合は、Integration Server Administrator からエイリアスを作成します。

- フロー、Java、C、アダプタまたは XSLT サービス以外のリソースにエイリアスを割り当てる。たとえば、Integration Server Administrator から、DSP、HTML、jpg、または Web サービスにエイリアスを関連付けることができます。
- 従来の方法を使用して設定された REST リソースの URL エイリアスを作成します。

  注:
  REST リソースの設定方法の詳細については、『*webMethods Service Development Help*』を参照してください。
- 特定のリソースに複数の URL エイリアスを関連付ける。
- 「空のパス」にエイリアスを作成する。
- 特定の URL エイリアスにポート固有の宛先を指定する。

サービスに単一の URL エイリアスを作成したい場合は、Designer の使用を検討してください。Designer を使用して URL エイリアスを作成するときは、URL パスを指定する必要はありません。Designer を使用して URL エイリアスを作成した場合は、エイリアスに割り当てられているパス名を更新しないでも、1 つのフォルダから別のフォルダにサービスを移動できます。Designer でのサービスへの URL エイリアス作成については、*webMethods Service Development Help*を参照してください。

HTTP URL エイリアスの作成

URL エイリアスを作成するときは、エイリアスと Integration Server のリソースの関連付けを行います。URL エイリアスを作成する場合は、以下の情報に留意してください。

- HTTP ポートまたは HTTPS ポートからの要求に対してのみ、URL エイリアスは作成できます。
- エイリアス名は、Integration Server 全体で一意である必要があります。
- ポートマッピングの指定により、単一 URL エイリアスを複数の宛先に関連付けることが可能です。要求を受信したポートに基づき、ポートマッピングはエイリアスを異なる URL エイリアスに相関させます。URLエイリアスのポートマッピング作成の詳細については、454 ページの「URL エイリアスでのポートマッピングの使用」を参照してください。
- localhost:5555 のような hostname:port の組み合わせのために、空のパス (/) の URL エイリアスを作成できます。空のパスエイリアスの作成により、任意の受信ポートに対してデフォルトの宛先を指定できます。空のパスエイリアスの詳細については、456 ページの「「空のパス」のための URL エイリアスの使用」を参照してください。
- REST リソースで URL エイリアスを使用したい場合は、URL エイリアスの部分一致を有効にする必要があります。詳細については、457 ページの「URL エイリアスの部分一致の有効化」を参照してください。
- URL エイリアスの部分一致を有効にする場合は、別のエイリアスで始まるエイリアスを定義しないでください。詳細については、457 ページの「URL エイリアスの部分一致の有効化」を参照してください。
- Designer または Integration Server を使用して、サービスの HTTP URL エイリアスを作成できます。しかし、複数のサービス、またはサービスとその他のリソースに対して同じ URL エイリアスを使用したい場合は、Integration Server Administrator を使用して URL エイリアスを作成する必要があります。Designer から HTTP URL エイリアスを作成する手順については、*webMethods Service Development Help*を参照してください。
- バージョン 10.3では、Microservices Runtime または、Microservices Runtime ライセンスを備えた Integration Server は、監視機能を提供する `health` および `metrics` エンドポイントと共に使用する「health」と「metrics」という名前の 2 つの事前定義済み URL エイリアスを作成します。Software AG は、これらと同じ名前の URL エイリアスを作成しないように推奨しています。また、Software AG は事前定義済みの「health」または「metrics」URL エイリアスを編集することをお勧めしません。

  注:
  Microservices Runtime の以前のバージョンからバージョン 10.3 以上に移行し、「health」と「metrics」の両方またはいずれか一方の名前の URL エイリアスが既に存在する場合、

Microservices Runtime は、「health」または「metrics」、もしくはその両方の URL エイリアスを作成しません。これにより、`health` エンドポイントまたは `metrics` エンドポイントの呼び出しの結果として、ヘルスインジケータまたはメトリクス収集が実行されないことがあります。ヘルスゲージとメトリクス収集の両方またはいずれか一方を使用する場合、「health」または「metrics」という名前の既存の URL エイリアスの名前を変更する必要があります。再起動すると、Microservices Runtime は、`health` エンドポイントまたは `metrics` 機能、もしくはその両方に対応する、新しい「health」または「metrics」、もしくはその両方の URL エイリアスを作成します。

HTTP URL エイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [URL エイリアス] に移動します。
3. [**URL エイリアスの作成**] をクリックします。
4. [**URL エイリアスのプロパティ**] で、以下の情報を指定します。

パラメータ	指定する値
[エイリアス]	エイリアスの名前。 エイリアスの名前にはスペース、および以下の文字を含むことができません。 # % ?' " < \ エイリアス名をストリング「http://」で始めることはできません。 エイリアスに長さ制限はありません。 **注:** 空のパスに URL エイリアスを使用する場合は、エイリアスを指定しないでください。
[空のパスエイリアスとして使用する]	空のパスにこのエイリアスを使用したい場合は、このチェックボックスをオンにしてください。 [**空のパスエイリアスとして使用する**] チェックボックスをオンにすると、Integration Server はエイリアスを <EMPTY> に変更します。
[デフォルト URL パス]	Integration Server 上のリソースへのパス。異なるタイプの宛先への URL パスの指定方法の詳細については、453 ページの「URL パスの指定」を参照してください。 URL エイリアスのポートマッピングを定義していない場合は、[**デフォルト URL パス**] フィールドを指定する必要があります。

パラメータ	指定する値
	URL エイリアスにポートマッピングが含まれている場合は、[**デフォルト URL パス**] フィールドはオプションです。しかし、**デフォルトのURLパス**値を与えると、ポートマッピングにないポートから要求を受信する場合にデフォルトの URL パスへ「誘導される」ので便利です。URL エイリアスがポートマッピングを指定しているが**デフォルトのURL パス**値を指定していない場合、Integration Server はそのエイリアスを URL パスとして使用します。 URL パスには空白および以下の文字は含めることができません: # % ? ' " < \
[パッケージ]	このエイリアスに関連付けるパッケージの名前。

5. 受信ポートに基づいて URL エイリアスに別の宛先を指定したい場合は、ポートマッピングを作成します。詳細については、454 ページの「URL エイリアスでのポートマッピングの使用」を参照してください。以下の手順に従います。

6. [**変更内容の保存**] をクリックします。

URL パスの指定

エイリアスが解決する先のリソースを定義します。URL エイリアスおよび/または、URL エイリアスのために定義されたポートマッピングの URL パスを指定します。

以下は、異なるリソースのための URL パスのフォーマット例です。

- リソースがサービスの場合

  invoke/*folder_name.subfolder_name/service_name*

- リソースが REST リソースの場合

  rest/*folder_name/subfolder_name*

- リソースが REST V2 リソースの場合

  restv2/*folder_name:resource_name*

- リソースがサービスでも REST リソースでもない場合

  package_name/file_name.ext

注:
URL パスには空白および以下の文字は含めることができません:

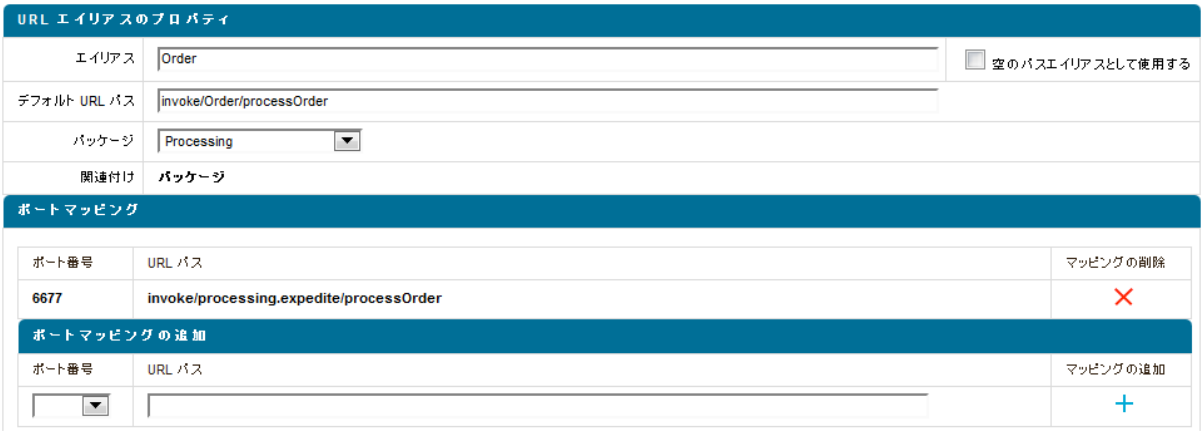
% ? ' " < \

URL エイリアスでのポートマッピングの使用

URL エイリアスには、ポートとリソースを関連付けるポートマッピングを 1 つ以上作成することができます。ポートマッピングを使用すると、ポートごとに異なる URL パスを設定できるので、複数の宛先を持つ単一の URL エイリアスを作成できます。ビジネスパートナごとに異なる宛先を定義したい場合があると思います。

Integration Server が URL エイリアスを含む要求を受信したとき、Integration Server は、まず、受信ポートに関連付けられた URL パスがあるかどうかを決定することにより要求を解決します。ポート番号にマッピングされた URL パスがない場合、Integration Server はそのエイリアスの要求先としてデフォルトの URL パスを使用します。常にポートマッピングが最初に検証されます。

例えば、以下の「Order」と命名された URL エイリアスで、「Order」というエイリアスを含む HTTP 要求をポート 6677 に受信すると、Integration Server は processing.expedite:processOrder というサービスを呼び出します。6677 以外のポートで受信した「Order」エイリアスを使用した HTTP 要求では、Integration Server は Order:processsOrder というサービスを呼び出します。



URL エイリアスへのポートマッピングの追加

URL エイリアスに対して複数のポートマッピングを作成できます。URL エイリアスのポートマッピングを定義する場合は、以下の点に留意してください。

- Integration Server Administrator で作成された URL エイリアスに対してのみポートマッピングを定義できます。Designer を使用してサービスのために定義された URL エイリアスにはポートマッピングは定義できません。
- ポートマッピングにないポートに受信した URL エイリアスを含む受信要求は、[**デフォルト URL パス**] フィールドに指定されているパスへ解釈されます。エイリアスが**デフォルトの URL パス**値を指定していない場合、Integration Server はそのエイリアスを URL パスとして使用します。

URL エイリアスにポートマッピングを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [URL エイリアス] に移動します。

3. HTTP URL エイリアスリストで、ポートマッピングを指定したい URL エイリアスを選択します。

4. ポートマッピングの [**ポート番号**] リストから、代替の宛先用に指定したいポートを選択します。Integration Server は、定義されている HTTP ポートおよび HTTPS ポートを一覧表示します。

5. [**URL パス**] フィールドに、宛先として使用したいリソースへのパスを指定します。URL パスには空白および以下の文字は含めることができません: # % ? ' " < \

   異なるリソースの指定方法の詳細については、453 ページの「URL パスの指定」を参照してください。

6. [マッピングの追加] で ＋ をクリックし、エイリアスにポートマッピングを追加します。

7. [**デフォルト URL パス**] フィールドではない別の宛先を指定したいポートごとに 4～6 の手順を繰り返します。

8. [**変更内容の保存**] をクリックします。

URL エイリアスからのポートマッピングの削除

URL エイリアスに追加したポートマッピングは削除できます。

注:
URL エイリアスのポートマッピングをすべて削除する場合は、必ず URL エイリアスに [**デフォルト URL パス**] 値を指定してください。

URL エイリアスのポートマッピングを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [URL エイリアス] に移動します。

3. HTTP URL エイリアスリストで、ポートマッピングを削除したい URL エイリアスを選択します。

4. [ポートマッピング] で、削除するポートマッピングの [削除] 列にある ✕ をクリックします。

5. エイリアスの削除したいポートマッピングごとに、上記手順を繰り返します。

6. [**変更内容の保存**] をクリックします。

「空のパス」のための URL エイリアスの使用

任意の HTTP/S ポートへの「空のパス」に対して URL エイリアスを作成できます。全 HTTP/S ポートが使用する宛先を 1 つ定義できます。または、各ポートにポートマッピングを追加することにより、任意の受信 HTTP/S ポート用の空のパスに対して異なるデフォルト宛先を定義できます。特定のポート用に空のパスに対してポートマッピングを定義し、残りの全ポートにデフォルトの宛先を定義することもできます。

Integration Server は以下のいずれかを空のパスとみなします。

- *host*:*port*
- *host*:*port*/

空のパスへの URL エイリアスには、<EMPTY> というエイリアス名が事前定義されています。各 Integration Server は <EMPTY> エイリアスを 1 つだけ持てます。

注:
Integration Server Administrator には URL 要求 *host*:*port*/WmRoot でアクセスできます。

空のパスの URL エイリアスの作成

空のパスエイリアスとして新しい URL エイリアスを作成するか、または Integration Server Administrator に作成された既存のエイリアスを空のパスエイリアスに変換します。Designer を使用してサービスのために作成された URL エイリアスは、空のパスエイリアスとして使用できません。

空のパスの URL エイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [URL エイリアス] に移動します。
3. 次のいずれかの手順に従います。
   - HTTP URL エイリアスリストで、空のパスエイリアスとして使用したい URL エイリアスを選択します。[**設定**] > [URL エイリアス] > [**エイリアス *aliasName* の編集**] ページで、[**空のパスエイリアスとして使用する**] チェックボックスをオンにします。Integration Server はそのエイリアス名を <EMPTY> に変更します。
   - [**URL エイリアスの作成**] をクリックします。[**設定**] > [URL エイリアス] > [エイリアスの作成] ページで、[**空のパスエイリアスとして使用する**] チェックボックスをオンにします。Integration Server はそのエイリアスを <EMPTY> に変更します。

     URL エイリアスのその他の設定情報を指定します。URL エイリアスの作成の詳細については、451 ページの「HTTP URL エイリアスの作成」を参照してください。
4. [**変更内容の保存**] をクリックします。

URL エイリアスの部分一致の有効化

場合によっては、URL 要求に特定のリソースに対する識別子が含まれることがあります。これらの識別子はリソースの各インスタンスによって異なるため、URL 要求は、特定のリソースに定義されたどの URL エイリアスとも完全に一致しない場合があります。そのようなリソースの URL エイリアスを定義できるようにするために、Integration Server は、部分一致を使用して URL 要求を処理できます。要求 URL が URL エイリアスの一部と一致するか、エイリアスの一部から始まる場合、***部分一致***となります。

部分一致が有効のときにIntegration Server が要求 URL を受信した場合、エイリアス全体が、要求 URL のパスの最初の文字から開始して、要求 URL の全部または最初の部分と一致すると、一致と見なされます。

たとえば、URL エイリアス a2 に rest/purchasing/invoice という URL パスがあり、Integration Server が次の要求 URL を受信したとします。

http://MyHost:5555/a2/75909

要求 URL は a2 エイリアスと一致し、パス: http://MyHost:5555/rest/purchasing/invoice/75909 に解釈されます。

Integration Server は、要求 URL の末尾文字を保持します。この場合、 Integration Server は /75909 を保持します。

注:
従来の方法を使用して設定された REST リソースで URL エイリアスを使用したい場合は、URL エイリアスの部分一致を有効にする必要があります。REST リソースの URL エイリアスの定義の詳細については、『*webMethods Service Development Help*』を参照してください。

Integration Server が URL 要求の部分一致を使用できるようにするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [拡張設定] に移動します。
3. [**拡張設定の編集**] をクリックします。
4. [**拡張設定**] ボックスに以下を追加します: `watt.server.url.alias.partialMatching=true`
5. [**変更内容の保存**] をクリックします。
6. Integration Server を再起動します。

HTTP URL エイリアスの表示

Integration Server で定義されているすべての HTTP URL エイリアスを表示するには、Integration Server Administrator から表示する必要があります。

すべての URL エイリアスのリストを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [URL エイリアス] に移動します。

HTTP URL エイリアスの関連付けについて

[設定] > [URL エイリアス] ページには、Integration Server Administrator から作成されたエイリアスと、Designer から作成されたエイリアスが表示されます。[**関連付け**] 列のコンテンツは、URL エイリアス作成に使用したツールを示します。

値	説明
[パッケージ]	エイリアスが Integration Server Administrator から作成されたことを示します。
service_name	エイリアスが Designer から作成されたことを示します。Designer からエイリアスを作成する場合、Designer または Integration Server にエイリアスを表示できます。さらに、エイリアスに割り当てられているパス名を更新せずに、1 つのフォルダから別のフォルダにサービスを移動できます。
[サーバ]	エイリアスが Integration Server の以前のバージョンで作成され、この Integration Server に移行されたものであることを示します。この段階でエイリアスは機能しますが、エイリアスに対して行える変更は、パッケージの関連付けの更新のみです。パッケージの関連付けを更新すると、他の HTTP URL エイリアスと同様にエイリアスを表示および変更できるようになります。

Designer からは、個別のサービスに割り当てられている HTTP URL エイリアスを表示できます。サービスのエイリアスが Designer または Developer から作成されたものである場合に限り、Designer でサービスのエイリアスが表示されます。

注:
HTTP URL エイリアスが以前のリリースの Integration Server から移行されたものである場合には、[HTTP URL エイリアスリスト] ページにエイリアスが表示されたときに、[**関連付け**] フィールドに「Server」が表示され [**パッケージ**] フィールドは空白になります。この段階でエイリアスは機能しますが、エイリアスに対して行える変更は、パッケージの関連付けの更新のみです。パッケージの関連付けを更新すると、他の HTTP URL エイリアスと同様にエイリアスを表示および変更できるようになります。

URL エイリアスの編集

Integration Server Administrator で作成された URL エイリアスのみ Integration Server Administrator を使用して編集できます。Designer で作成された URL エイリアスは編集できません。

URL エイリアスを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [URL エイリアス] に移動します。
3. [HTTP URL エイリアス] リストから、編集したい URL エイリアスを選択します。
4. URL エイリアスに変更を加えます。
5. [**変更内容の保存**] をクリックします。

URL エイリアスの削除

Integration Server Administrator または Designer で作成された URL エイリアスのみ Integration Server Administrator を使用して削除できます。

注:
パッケージを削除または無効にすると、Integration Server Administrator によって自動的に、パッケージに関連付けられているすべての URL エイリアスが削除されます。

URL エイリアスを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [URL エイリアス] に移動します。
3. 削除するエイリアスが含まれている行を見つけて、✕ アイコンをクリックします。

   エイリアスを削除するかどうかを確認するダイアログが Integration Server で表示されます。
4. [**OK**] をクリックします。

URL エイリアスの移植性と競合の可能性

Designer では、1 つのフォルダから別のフォルダにサービスを移動できます。このとき、エイリアスはサービスと一緒に移動します。

Integration Server で、Integration Server のパッケージの複製機能を使用して 1 つの Integration Server から別の Integration Server にパッケージをコピーすると、パッケージに関連付けられている HTTP URL エイリアスもパッケージと一緒にコピーされます。エイリアスがパッケージと一緒にコピーされることは、エイリアスが Integration Server で作成されたか Designer または Developer で作成されたかに関係なく当てはまります。

別の Integration Server からパッケージを受信した場合に、新しいパッケージに関連付けられているエイリアスの名前が、Integration Server に既に定義されている HTTP URL エイリアスの名前と同じである可能性があります。

Integration Server では HTTP URL エイリアス名のレジストリが保持されており、[**設定**] > [URL エイリアス] ページにこのリストの内容が表示されます。Integration Server は、サーバ起動時にパッケージをロードするとき、およびエイリアスが追加、編集または削除されたときにこのリストを作成します。2つのパッケージが同じエイリアス名に関連付けられている場合、Integration Server はロードする 2 つのパッケージのうちの 1 つ目に関連付けられているエイリアスを使用します。2 つ目のパッケージをロードする際に、Integration Server はそのパッケージに関連付けられているエイリアスに重複のフラグを立てます。レジストリには追加しません。

つまり、Integration Server はパッケージのロード順序を使用して、優先権を決定します。WmRoot は常に最初にロードされます。他のパッケージのロード順序は、パッケージの依存性やオペレーティングシステムなど、その他の要因によるため、簡単には予測できません。

そのため、別の Integration Server でパブリッシュされたパッケージをインストールする場合には、パッケージの詳細ページ ([パッケージ] > [管理] > 「*package_name*」) でロードの警告をチェックしてください。

23 HTTP インターセプタの使用

- HTTP インターセプタの概要 462
- HTTP インターセプタの作成に関する考慮事項 463
- HTTP インターセプタの作成 464

HTTP インターセプタの概要

Integration Server は HTTP 要求と HTTP 応答を取得し処理する HTTP インターセプタの作成フレームワークを備えています。HTTP インターセプタは HTTP 要求および HTTP 応答への読み取り専用アクセスを行います。HTTP インターセプタはすべての HTTP 要求および HTTP 応答を取得できますが、例外は、OAuth メッセージ、独自の HTTP URL 接続を使用するサービス、および NTLM、DIGEST、REDIRECTS の媒介要求と応答です。

HTTP インターセプタのタイプ

2 つのタイプの HTTP インターセプタがあります。

- 受信インターセプタ。Integration Server が受信 HTTP 要求を処理するときに呼び出されます。ここで、Integration Server は HTTP サーバです。
- 送信インターセプタ。Integration Server が送信 HTTP 要求を送信するときに呼び出されます。ここで、Integration Server は HTTP クライアントです。

インターセプタのどちらか一方または両方のタイプを実装して、データレイクまたはログファイルに HTTP 要求を書き込むなどのタスクを実行できます。

HTTP インターセプタのインタフェース

HTTP インターセプタを作成するには、次のインタフェースの 1 つを実装する Java クラスを作成します。

- 受信インターセプタの場合: com.softwareag.is.interceptor.HttpInterceptorIFC
- 送信インターセプタの場合: com.softwareag.is.interceptor.HttpOutboundInterceptorIFC

両方のインタフェースでは、実装元クラスが要求と応答の処理用に実装できる 2 つのメソッドを宣言します。

- `preProcess` メソッド。HTTP 要求に対して呼び出されます。受信インターセプタの場合、Integration Server で受信された HTTP 要求に対して呼び出されます。送信インターセプタの場合、Integration Server で送信された HTTP 要求に対して呼び出されます。
- `postProcess` メソッド。HTTP 応答に対して呼び出されます。受信インターセプタの場合、Integration Server で送信された HTTP 応答に対して呼び出されます。送信インターセプタの場合、Integration Server で受信された HTTP 応答に対して呼び出されます。

HTTP インターセプタの機能

Integration Server は、HTTP ヘッダー、ステータスコード、ステータスメッセージを含む、未加工の HTTP 要求と応答に HTTP インターセプタがアクセスできるようにします。

受信インターセプタの機能は次のとおりです。

- 要求: Integration Server は、最初にクライアントを認証した後、受信 HTTP インターセプタを呼び出します。受信インターセプタの `preProcess` メソッドはヘッダーを含む未加工の HTTP 要求データおよび要求本文のクローンにアクセスできます。`preProcess` メソッドは、Integration Server が要求のコンテンツハンドラの使用などの処理を行う前に呼び出されます。必要に応じて、`preProcess` メソッドは HttpInterceptorException をスローして、要求の通常処理を停止できます。受信インターセプタの場合、この例外には、Integration Server が要求元クライアントへの応答として送信する HTTP ステータスコードおよびメッセージが含まれます。
- 応答: 受信 HTTP インターセプタは、Integration Server がクライアントに送信する直前に HTTP 応答を取得します。Integration Server が応答を接続に書き込む前に、受信インターセプタの `postProcess` メソッドは、応答本文のクローン、応答ステータスコード、ステータスメッセージ、およびヘッダーにアクセスできます。

送信インターセプタの機能は次のとおりです。

- 要求: Integration Server は、要求をプロバイダに送信する直前に、送信 HTTP インターセプタの `preProcess` メソッドを呼び出します。`preProcess` メソッドはヘッダーを含む未加工の HTTP 要求データおよび要求本文のクローンにアクセスできます。必要に応じて、`preProcess` メソッドは HttpInterceptorException をスローして、Integration Server が要求を送信しないようにできます。
- 応答: Integration Server は受信する応答のクローンを作成し、送信インターセプタの `postProcess` メソッドを呼び出します。`postProcess` メソッドは Integration Server 応答処理と並行してクローン化された応答を処理します。

HTTP インターセプタの作成に関する考慮事項

HTTP インターセプタの作成時には、以下の点を考慮する必要があります。

- HTTP インターセプタクラスは、受信メッセージ用の com.softwareag.is.interceptor.HttpInterceptorIFC インタフェースおよび送信メッセージ用の com.softwareag.is.interceptor.HttpOutboundInterceptorIFC を実装する必要があります。
- 各タイプの HTTP インターセプタクラスを 1 つだけ登録できます。
- HTTP インターセプタクラスは `preProcess` メソッドおよび `postProces` メソッドの実装を提供する必要があります。詳細については、『*webMethods Integration Server Java API Reference*』を参照してください。
- HTTP インターセプタを使用すると、要求の処理に時間がかかります。要求処理を続行するかどうかを決定するために `preProcess` メソッドが実行するロジックは、同期的に実行する必要があります。しかし、別スレッドを発生させることで非同期に実行できるロジックが `preProcess` メソッドに含まれている場合、通常処理を遅延なしに続行できます。
- また、HTTP インターセプタを使用すると、応答の処理にも時間がかかります。
- HTTP インターセプタは HTTP 要求または応答の内容を変更できません。
- 受信インターセプタの場合、Integration Server は入力ストリーム (送信インターセプタの場合は出力ストリーム) のバイト配列表現を渡します。

注:
送信インターセプタによる処理に必要なペイロードに対して十分なシステムリソースがあることを確認してください。

- HTTP インターセプタが HttpInterceptorException をスローして要求を拒否しない限り、Integration Server は元の入力 (または出力) ストリームの通常処理を続行します。

HTTP インターセプタの作成

下記の手順を使用して、一方または両方のインターセプタクラスを作成し、追加できます。

HTTP インターセプタを作成し追加するには

1. 必要な (HttpInterceptorIFC または HttpOutboundInterceptorIFC) インタフェースを実装する Java クラスを作成します。
2. クラスを作成し、jar ファイルに追加します。
3. Integration Server がクラスパスに追加する次のいずれかの場所に jar ファイルを配置します。
   - *Software AG_directory* \ *Integration Server_directory* \instances*instanceName*\lib\jars\custom
   - *Software AG_directory* \ *Integration Server_directory* \lib\jars\custom
   - *Software AG_directory* \ *Integration Server_directory* \instances*instanceName*\packages*packageName*\code\jars\static
4. Integration Server で以下のサーバ設定プロパティを設定します。
   - 受信インターセプタの場合
     - `watt.server.http.interceptor.enabled`: 受信 HTTP インターセプタを使用するには、このパラメータを「true」に設定します。
     - `watt.server.http.interceptor.impl`: 受信インターセプタインタフェースを実装するクラスの完全修飾名にこのパラメータを設定します。
     - `watt.server.http.interceptor.preprocess.sizeLimit`: 受信インターセプタによって処理される要求にサイズ制限を定義するには、このパラメータを設定します。このパラメータは送信インターセプタには適用できません。
   - 送信インターセプタの場合
     - `watt.server.http.interceptor.outbound.enabled`: 受信 HTTP インターセプタを使用するには、このパラメータを「true」に設定します。
     - `watt.server.http.interceptor.outbound.impl`: 送信インターセプタインタフェースを実装するクラスの完全修飾名にこのプロパティを設定します。

以上のパラメータの詳細については、1136 ページの「watt.server.」を参照してください。

5. Integration Server を再起動します。

24 Integration Server の HTTP 圧縮の設定

- 概要 468
- HTTP 要求の圧縮 469
- HTTP 応答の圧縮 470

概要

Integration Server では、次の場合に HTTP 要求および HTTP 応答に対するデータ圧縮 (コンテンツエンコーディングとも呼びます) をサポートしています。

- Integration Server が HTTP クライアントとして機能しており、HTTP 要求で圧縮されたペイロードを送信する場合
- Integration Server が HTTP クライアントとして機能しており、HTTP 応答で圧縮されたペイロードを受信する場合
- Integration Server が HTTP サーバとして機能しており、HTTP 要求で圧縮されたペイロードを受信する場合
- Integration Server が HTTP サーバとして機能しており、HTTP 応答で圧縮されたペイロードを送信する場合

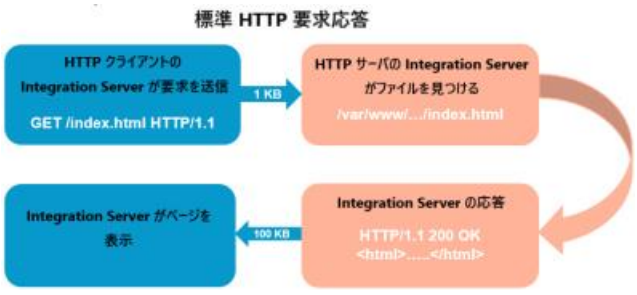
HTTP 圧縮スキームによって、データ転送速度が向上し、ネットワーク帯域幅は効率よく利用されます。ユーザが大量のデータを持っており、HTTP 要求によってそのデータを送信しようとしている場合、Integration Server では送信前のデータの圧縮をサポートします。また、HTTP 応答の送信前のデータの圧縮もサポートします。

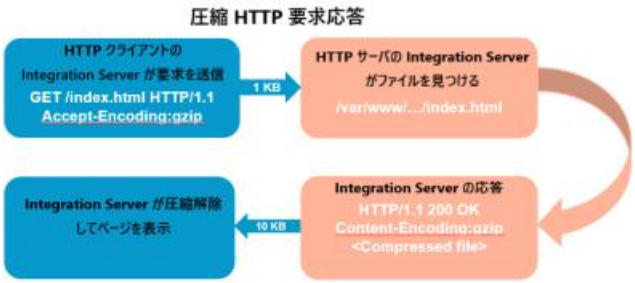
Integration Server では、次の HTTP 圧縮スキームをサポートしています。

- Gzip
- Deflate

注:
Integration Server では、ダイナミックサーバページ (DSP) の HTTP 要求または応答の圧縮をサポートしていません。

HTTP クライアントとしての Integration Server は HTTP 要求の送信前のデータ圧縮をサポートします。また、クライアントに応答を返す前にデータを圧縮するよう HTTP サーバに指示することもできます。





Integration Server が HTTP クライアントとして機能している場合、次の組み込みサービスを使用できます。

- pub.compress:compressData。HTTP 要求の送信前にデータを圧縮します。
- pub.compress:decompressData。HTTP 応答の受信後にデータを圧縮解除します。

これらのサービスの詳細については、『*webMethods Integration Server Built-In Services Reference* ガイド』を参照してください。

Integration Server が HTTP サーバとして機能している場合、次のサーバ設定パラメータを「`true`」に設定する必要があります。

- `watt.server.http.request.supportCompression`。HTTP 要求の圧縮をサポートします。
- `watt.server.http.response.supportCompression`。HTTP 応答の圧縮をサポートします。

このパラメータの詳細については、1136 ページの「watt.server.」を参照してください。

HTTP 要求の圧縮

Integration Server が HTTP クライアントとして機能しており、ユーザに圧縮する大量のデータセットがある場合、ユーザは pub.compress:compressData サービスを使用してデータを圧縮できます。サービスの実行中に、ユーザはサポートされた圧縮スキームを定義してデータを圧縮できます。クライアントは、HTTP 要求を送信する場合に、`Content-Encoding` HTTP ヘッダーを `gzip` または `deflate` と定義する必要があります。このヘッダーは、クライアントが要求ペイロードの圧縮に使用した圧縮スキームを示します。

Integration Server が HTTP サーバとして機能し、サーバ設定パラメータ `watt.server.http.request.supportCompression` が `true` に設定されている HTTP 圧縮をサポートしている場合、HTTP 要求の受信後、Integration Server は `Content-Encoding` に記述された圧縮スキームと同じスキームを使用してデータを圧縮解除した後、さらに処理するためデータを送信します。

HTTP 要求の `Content-Encoding` に複数のアルゴリズムスキームが含まれている場合、HTTP サーバは最初に記述された有効な圧縮スキームを使用します。複数のアルゴリズムスキームを含む HTTP ヘッダーの以下の例について考えます。

```
Content-Encoding: deflate, gzip
```

この場合、Integration Server は deflate 圧縮スキームを使用します。

```
Content-Encoding: <UnSupportedType>, gzip
```

この場合、Integration Server は gzipe 圧縮スキームを使用します。

```
Content-Encoding: deflate, <UnSupportedType>
```

この場合、Integration Server は deflate 圧縮スキームを使用します。

```
Content-Encoding: <UnSupportedType>, <UnSupportedType>
```

この場合、Integration Server は HTTP 要求を無視します。

HTTP 要求圧縮の例です。

```
Request
POST http://localhost:<port>/restv2/testEncoding:ResourceForEncoding/testRequest
Content-Encoding: gzip
Response
HTTP/1.1 200 OK
```

HTTP 応答の圧縮

Integration Server が HTTP クライアントとして機能しており、クライアントが HTTP サーバからの応答として大量のデータセットを想定している場合、クライアントへ応答を返す前にデータを圧縮するよう、クライアントはサーバに要求できます。`Accept-Encoding` HTTP ヘッダーが `gzip` または `deflate` に設定された HTTP 要求をクライアントが送信する場合、これはサーバが応答ペイロードの圧縮に使用できる圧縮スキームを示します。

Integration Server が HTTP サーバとして機能し、サーバ設定パラメータ `watt.server.http.response.supportCompression` が `true` に設定されている HTTP 圧縮をサポートしている場合、HTTP 応答の送信中に Integration Server は `Content-Encoding` HTTP ヘッダーを `gzip` または `deflate` と定義します。このヘッダーは、サーバがデータの圧縮に使用した圧縮スキームを示します。

HTTP クライアントとして機能している Integration Server がサーバから圧縮された応答を受信し、ユーザは pub.compress:decompressData サービスを使用してデータを圧縮解除できます。このサービスは、HTTP ヘッダーに記述された圧縮スキームと同じスキームを使用してデータを圧縮解除します。

HTTP 要求の `Accept-Encoding` に複数のアルゴリズムスキームが含まれている場合、HTTP サーバは最初に記述された有効な圧縮スキームを使用します。複数のアルゴリズムスキームを含む HTTP ヘッダーの以下の例について考えます。

```
Accept-Encoding: deflate, gzip
```

この場合、Integration Server は deflate 圧縮スキームを使用します。

```
Accept-Encoding: <UnSupportedType>, gzip
```

この場合、Integration Server は gzipe 圧縮スキームを使用します。

```
Accept-Encoding: deflate, <UnSupportedType>
```

この場合、Integration Server は deflate 圧縮スキームを使用します。

```
Accept-Encoding: <UnSupportedType>, <UnSupportedType>
```

この場合、Integration Server は圧縮されていないデータで応答します。

HTTP 応答圧縮の例です。

```
Request
GET http://localhost:<port>/rad/testEncoding:ResourceForEncoding/testResponse
Accept-Encoding: gzip
Response
HTTP/1.1 200 OK
Content-Encoding: gzip

Request
GET http://localhost:<port>/rad/testEncoding:ResourceForEncoding/testResponse
Accept-Encoding: deflate
Response
HTTP/1.1 200 OK
Content-Encoding: deflate

Request
GET http://localhost:<port>/invoke/folder:serviceName
Accept-Encoding: gzip, deflate
Response
HTTP/1.1 200 OK
Content-Encoding: gzip

Request
GET http://localhost:<port>/odata/folder:odataEnityName/Resource
Accept-Encoding: deflate, gzip
Response
HTTP/1.1 200 OK
Content-Encoding: deflate
```

25 Integration Server の SFTP サーバへの接続の設定

- SFTP の概要 474
- SFTP サーバエイリアスの作成 474
- SFTP ユーザエイリアスの作成 477

SFTP の概要

SSH ファイル転送プロトコル (SFTP) は、セキュアシェルプロトコル (SSH) に基づくネットワークプロトコルです。SFTP を使用すると、信頼性の高い任意のデータストリームを介した、セキュアなファイルアクセス、ファイル転送およびファイル管理容易になります。

Integration Server は、SFTP プロトコルを使用して SFTP サーバに接続し、以下のタスクを実行するように設定することができます。

- Integration Server と SFTP サーバ間のファイル転送。SFTP サーバからファイルを取得してローカルマシンに保存したり、ローカルマシンから SFTP サーバにファイルをアップロードしたりできます。
- SFTP サーバ内のファイルへのアクセス。SFTP サーバ内のディレクトリやファイルを表示したり、その権限および所有権情報を表示したりできます。
- SFTP サーバ内のディレクトリまたはファイルの管理。SFTP サーバ内でファイルまたはディレクトリを作成、名前変更または削除できます。SFTP サーバ内のファイルの権限または所有権を変更することもできます。

Integration Server Administrator を使用すると、以下の SFTP エイリアスを定義できます。

- **SFTP サーバエイリアス**SFTP サーバエイリアスには、SFTP サーバに接続するために Integration Server が使用する設定パラメータが含まれます。
- **SFTP ユーザエイリアス**SFTP ユーザエイリアスには、SFTP クライアントとしての認証および機能のために Integration Server が使用するクライアント設定パラメータが含まれます。

SFTP サーバエイリアスの作成

SFTP サーバエイリアスは、名前が付けられた一連のパラメータであり、Integration Server が、サーバに接続するために使用します。

重要:
SFTP ユーザエイリアスを作成する前に、少なくとも 1 つの SFTP サーバエイリアスを作成する必要があります。

SFTP サーバエイリアスを作成するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [SFTP] に移動します。
3. [**サーバエイリアスの作成**] をクリックします。
4. 以下のサーバエイリアス設定を指定します。

パラメータ	指定内容
[SFTP サーバエイリアスのプロパティ]	
[SFTP クライアントバージョン]	使用する SFTP クライアント。 **注:** バージョン 2 クライアントには、追加の設定プロパティ、キー交換アルゴリズム、MAC (Machine Access Code: マシンアクセスコード) アルゴリズム、およびバージョン 1 クライアントで使用できなかった暗号が含まれています。
[エイリアス]	SFTP サーバエイリアスの名前。 SFTP サーバエイリアスの名前は以下のようになります。 ■ 特殊文字はアンダースコア (_) およびピリオド (.) のみを使用できます。 ■ ストリング「http://」で始めることはできません。 ■ 最大長は 255 文字です。
[ホスト名または IP アドレス]	SFTP サーバのホスト名または IP アドレス。
[ポート番号]	SFTP サーバのポート番号。ポート番号の範囲は 0 以上 65535 以下とする必要があります。
[プロキシエイリアス]	要求をルーティングするときに経由するプロキシエイリアス。プロキシエイリアスには、HTTP、HTTPS または SOCKS を指定できます。プロキシエイリアスが指定されていない場合、Integration Server では、要求の送信に成功するか、すべてのプロキシサーバを試すまで、有効なプロキシサーバエイリアスを 1 つずつ使用して送信要求を繰り返します。プロキシエイリアスの詳細については、140 ページの「プロキシサーバエイリアスの作成」を参照してください。
[ホストキーの場所]	SFTP サーバの公開鍵の場所。Integration Server はその公開鍵を使用して SFTP サーバの妥当性検査を行います。 **重要:** 公開鍵ファイルは、Integration Server をインストールしたマシンと同じマシン上に存在する必要があります。

パラメータ	指定内容
	SFTP サーバの公開鍵がない場合は、[**ホストキーを取得**] をクリックします。Integration Server は、指定したホストおよびポートの公開鍵を抽出し、それを一時フォルダに保存します。次に、Integration Server は [**ホストキーの場所**] フィールドの一時フォルダパスを表示します。
[SFTP サーバエイリアスの拡張設定 (オプション)]	
[最小 DH キーサイズ]	最小 DH キーサイズ。このパラメータは SFTP クライアントバージョン 1 には適用できません。
[最大 DH キーサイズ]	最大 DH キーサイズ。このパラメータは SFTP クライアントバージョン 1 には適用できません。
[優先キー交換アルゴリズム]	キー交換のために Integration Server が SFTP サーバに提示するキー交換アルゴリズムのリスト。アルゴリズムは優先順位に従って表示されます。
[優先される MAC アルゴリズム S2C]	SFTP サーバから Integration Server に送信されるメッセージに適用できるメッセージ認証コード (MAC) アルゴリズムのリスト。アルゴリズムは優先順位に従って表示されます。
[優先される MAC アルゴリズム C2S]	Integration Server から SFTP サーバに送信されるメッセージに適用できるメッセージ認証コード (MAC) アルゴリズムのリスト。アルゴリズムは優先順位に従って表示されます。
[優先される Ciphers S2C]	SFTP サーバから Integration Server に送信されるメッセージに適用できる、優先される暗号のリスト。
[優先される Ciphers C2S]	Integration Server から SFTP サーバに送信されるメッセージに適用できる、優先される暗号のリスト。

- Integration Server がアルゴリズムを SFTP サーバに提示する順序を指定するには、アルゴリズムを選択し、[**上**] または [**下**] をクリックします。
- すべての SFTP サーバはアルゴリズムと暗号の事前定義済みセットをサポートします。Integration Server および SFTP サーバの両方がサポートするアルゴリズムが選択されます。
- リストからアルゴリズムを除外するには、アルゴリズムを選択し、[**右**] をクリックします。アルゴリズムは除外されるアルゴリズムのリストに移動します。

詳細については、481 ページの「既存のサーバエイリアスデータに対するアップグレードの影響」を参照してください。

5. [**変更内容の保存**] をクリックします。

注:
エラーメッセージ「**com.maverick.ssh.SshException: Minimum DH p value not provided [2048]**」が表示された場合、[**最小 DH キーサイズ**] を 2048 に設定してください。

注:
`diffie-hellman-group-exchange-sha1`、`diffie-hellman-group-exchange-sha256` などのキー交換アルゴリズムでは、[**最小 DH キーサイズ**] および [**最大 DH キーサイズ**] パラメータに設定された値が考慮されます。他のすべてのアルゴリズムではこれらの値は無視されます。

Integration Server は、SFTP サーバエイリアス設定とホストキー情報を `/<IntegrationServer_directory>/instances/<instance_name>/config/sftp/sftpServerAliases.cnf` ファイルに保存します。

SFTP サーバエイリアスの編集

編集できるプロパティは、[**ホスト名または IP アドレス**]、[**ポート番号**] および [**ホストキーの場所**] のみです。[**ホスト名または IP アドレス**] および [**ポート番号**] のフィールドを空にすることはできません。

SFTP サーバエイリアスを編集するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [SFTP] に移動します。
3. [SFTP サーバリスト] で、編集する SFTP サーバエイリアスの名前をクリックします。
4. 選択したエイリアスのプロパティページで、必要な変更を加えます。
5. [**変更内容の保存**] をクリックします。

SFTP ユーザエイリアスの作成

SFTP ユーザエイリアスは、名前が付けられた一連のパラメータであり、Integration Server が、SFTP クライアントとして機能するために使用する、SFTP ユーザアカウントの詳細およびクライアント設定が含まれます。

多くの組織では、SFTP ユーザエイリアスの作成に必要な SFTP ユーザアカウント情報は、システム管理者が提供します。

同じ SFTP ユーザアカウントに対して、複数の SFTP ユーザエイリアスを設定することができます。Integration Server 内の各 SFTP ユーザエイリアス名は、一意とする必要があります。

SFTP クライアントとして機能するように Integration Server を設定する場合は、以下の点に留意してください。

- Integration Server は、自身をクライアントとして SFTP サーバに認証するため、パスワード認証および公開鍵認証をサポートしています。
- パスワード認証および公開鍵認証のどちらの場合も、SFTP アクセス用に設定されたアカウントを SFTP サーバ上に用意する必要があります。
- 公開鍵認証の場合、SFTP サーバおよび Integration Server は、自身の秘密鍵および互いの公開鍵にアクセスできる必要があります。

重要:
SFTP ユーザエイリアスを作成する前に、少なくとも 1 つの SFTP サーバエイリアスを作成する必要があります。

SFTP ユーザエイリアスを作成するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [SFTP] に移動します。
3. [**ユーザエイリアス設定**] をクリックします。
4. [**ユーザエイリアスの作成**] をクリックします。
5. [**SFTP ユーザエイリアスのプロパティ**] で、以下の情報を指定します。

パラメータ	指定内容
[エイリアス]	エイリアスの名前。 SFTP ユーザエイリアスの名前は以下のようになります。 ■ 特殊文字はアンダースコア (_) およびピリオド (.) のみを使用できます。 ■ ストリング「http://」で始めることはできません。 ■ 最大長は 255 文字です。
[ユーザ名]	SFTP ユーザアカウントのユーザ名。
[認証タイプ]	SFTP サーバに対して自身を認証するために Integration Server が使用する認証のタイプ。クライアント認証には、パスワードまたは公開鍵と秘密鍵のいずれかを使用できます。 **選択項目　目的**

パラメータ	指定内容
	[パスワード] パスワード認証を使用します。 **[公開鍵]** 公開鍵および秘密鍵を使用して Integration Server を認証します。 この認証タイプを使用する場合、SFTP サーバおよび Integration Server はそれぞれ、自身の秘密鍵および互いの公開鍵にアクセスできる必要があります。公開鍵認証を選択した場合、Integration Server は秘密鍵ファイルを *Integration Server_directory* /instances/*instance_name*/config/sftp/identities フォルダに保存します。
[パスワード]	パスワード認証を使用する場合は、SFTP サーバに接続する指定したユーザのパスワードを入力します。
[パスワードの再入力]	上記のパスワードを再入力します。
[秘密キーの場所]	認証タイプとして **[公開鍵]** を選択した場合は、指定した SFTP ユーザの秘密キーの場所を入力します。
[パスフレーズ]	認証タイプとして **[公開鍵]** を選択し、指定した秘密鍵にパスフレーズが必要な場合は、指定したユーザの秘密鍵ファイルのパスフレーズを入力します。
[パスフレーズの再入力]	上記のパスフレーズを再入力します。
[SFTP サーバエイリアス]	指定したユーザで接続する SFTP サーバのエイリアス。
[最大再試行回数]	Integration Server が SFTP サーバへの接続を試行する回数。許容される最大値は 6 です。許容される最小値は 1 です。デフォルトの試行回数は 6 回です。 **注:** このフィールドは SFTP クライアントのバージョン 1 で作成されたユーザエイリアスにのみ適用できます。
[接続タイムアウト (秒)]	タイムアウトして要求を終了するまでに SFTP サーバからの応答を Integration Server が待機する時間 (ミリ秒)。デフォルトは 0 で、これはセッションがタイムアウトしないことを示します。
[セッションのタイムアウト (分)]	アイドルセッションを終了するまでの Integration Server の待機時間。デフォルトは 10 分です。
[厳格なホストキー検査]	Integration Server で、SFTP サーバへの接続を確立する前に、SFTP サーバのホストキーを検査するかどうか。

パラメータ	指定内容
	選択項目 / **目的** **[はい]** Integration Server で SFTP サーバエイリアスの設定中にインポートされたホストキーに対して SFTP サーバのホストキーの検査を許可します。ホストキーが同じである場合、Integration Server では SFTP サーバへの接続が確立されます。同じではない場合、SFTP サーバへの接続は失敗します。 **[いいえ]** 接続中に Integration Server での SFTP サーバのホストキーの検査を許可しません。 **注:** セキュリティを強化するため、Software AG では **[はい]** を選択し、ホストキー検査を有効にすることをお勧めします。
[圧縮]	送信されるデータ量を減らすためにデータを圧縮するかどうか。Integration Server は、圧縮アルゴリズム zlib を使用した圧縮をサポートしています。 **注:** 圧縮を使用できるのは、接続先の SFTP サーバが圧縮をサポートしている場合のみです。 **選択項目** / **目的** **[zlib]** SFTP サーバと Integration Server の間で転送されるデータを圧縮します。 **[none]** データを圧縮しません。
[圧縮レベル]	**[圧縮]** フィールドで圧縮アルゴリズム zlib を選択した場合に使用する圧縮レベル。許容される最小値は 1 です (高速、低圧縮)。許容される最大値は 6 です (低速、高圧縮)。デフォルトは 6 です。 **注:** このフィールドは SFTP クライアント認証のバージョン 1 で作成されたユーザエイリアスにのみ適用できます。

6. **[変更内容の保存]** をクリックします。

   Integration Server は、SFTP ユーザエイリアス設定を *Integration Server_directory* /instances/*instance_name*/config/sftp/sftpUserAliases.cnf ファイルに保存します。

SFTP ユーザエイリアスの編集

SFTP ユーザアカウントのエイリアス名およびユーザ名を除き、すべてのフィールドを編集できます。

SFTP ユーザエイリアスを編集するには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [SFTP] > [ユーザエイリアスの設定] に移動します。
3. [SFTP ユーザリスト] で、編集する SFTP ユーザエイリアスの名前をクリックします。Integration Server に、そのエイリアスのプロパティページが表示されます。
4. 選択したエイリアスのプロパティページで、必要な変更を加えます。
5. [**変更内容の保存**] をクリックします。

SFTP 設定における移行の影響

Integration Server 9.12 より前には、[**優先キー交換アルゴリズム**] および [**プロキシエイリアス**] フィールドは SFTP ユーザエイリアスで指定されました。これらのフィールドは SFTP サーバエイリアスで指定されるようになりました。以前のバージョンから Integration Server 9.12 以降に移行する場合、Integration Server は [**優先キー交換アルゴリズム**] および [**プロキシエイリアス**] フィールドの値を以下のように決定します。

- SFTP サーバエイリアスが SFTP ユーザエイリアスで使用されたことがなかった場合、Integration Server は [**優先キー交換アルゴリズム**] にデフォルトの順序、[**プロキシエイリアス**] にデフォルト値 `None` を使用します。
- SFTP サーバエイリアスが 1 つの SFTP ユーザエイリアスでのみ使用された場合、Integration Server は [**優先キー交換アルゴリズム**] の順序および [**プロキシエイリアス**] の値を SFTP ユーザエイリアスから SFTP サーバエイリアスに移行します。
- SFTP サーバエイリアスが複数の SFTP ユーザエイリアスで使用された場合、Integration Server は [**優先キー交換アルゴリズム**] の順序および [**プロキシエイリアス**] の値を SFTP サーバエイリアスに関連付けられた最初の SFTP ユーザエイリアスから移行します。

既存のサーバエイリアスデータに対するアップグレードの影響

Integration Server 10.7 にアップグレードする場合、既存の SFTP サーバエイリアスに対する SFTP クライアントバージョンはバージョン 1 に設定されます。さらに、以下のように、他のバージョン 1 のフィールドに watt.ssh.jsch.* プロパティの値が入力されます。

- [**優先されるキー交換アルゴリズム**]: `watt.ssh.jsch.kex` サーバプロパティの値に含まれるアルゴリズムが [優先されるキー交換アルゴリズム] に追加され、他のすべてのアルゴリズムが

[除外されるキー交換アルゴリズム] に追加されます。watt プロパティの値が空である場合、デフォルトのキー交換アルゴリズムが [優先されるキー交換アルゴリズム] に追加されます。

- [**優先される MAC アルゴリズム S2C**]: `watt.ssh.jsch.mac_s2c` サーバプロパティの値に含まれるアルゴリズムが [優先される MAC アルゴリズム S2C] に追加され、他のすべてのアルゴリズムが [除外される MAC アルゴリズム S2C] に追加されます。watt プロパティの値が空である場合、デフォルトのサーバからクライアントへの MAC アルゴリズムが [優先される MAC アルゴリズム S2C] に追加されます。
- [**優先される MAC アルゴリズム C2S**]: `watt.ssh.jsch.mac_c2s` サーバプロパティの値に含まれるアルゴリズムが [優先される MAC アルゴリズム C2S] に追加され、他のすべてのアルゴリズムが [除外される MAC アルゴリズム C2S] に追加されます。watt プロパティの値が空である場合、デフォルトのクライアントからサーバへの MAC アルゴリズムが [優先される MAC アルゴリズム C2S] に追加されます。
- [**優先される Ciphers S2C**]: `watt.ssh.jsch.ciphers` サーバプロパティの値に含まれる暗号が [優先される Ciphers S2C] に追加され、他のすべての暗号が [除外される Ciphers S2C] に追加されます。
- [**優先される Ciphers C2S**]: `watt.ssh.jsch.ciphers` サーバプロパティの値に含まれる暗号が [優先される Ciphers C2S] に追加され、他のすべての暗号が [除外される Ciphers C2S] に追加されます。

注:

- `watt.ssh.jsch.logging` を除くすべての `watt.ssh.jsch.*` パラメータは廃止されています。廃止されたパラメータを使用しないでください。優先されるキー交換アルゴリズム、暗号、および MAC アルゴリズムはユーザインタフェースから設定されるためです。
- Integration Server は `watt.ssh.jsch.logging` サーバ設定プロパティを使用して、SFTP クライアントの両方のバージョンのログを有効にします。

SFTP サーバへの接続のテスト

SFTP ユーザエイリアスを追加した後、接続をテストすると、エイリアスに指定したクレデンシャルおよび詳細を使用して Integration Server が SFTP サーバとの接続を確立できることを確認できます。

SFTP サーバへの接続をテストするには

1. Integration Server Administrator を開きます。
2. [**外部サーバ**] > [SFTP] > [ユーザエイリアスの設定] に移動します。

   Integration Server Administrator に、すべての SFTP ユーザエイリアスの定義が表示されます。
3. テストするエイリアスの [テスト] 列にある ▶ アイコンをクリックします。

   Integration Server Administrator に、正しく接続されているかどうかを示す状態行が表示されます。

26 Integration Server との通信のセキュリティの設定

- Integration Server における SSL の概要 484
- サポートされている SSL/TLS プロトコル 485
- SSL と使用する暗号スイートの指定 488
- Integration Server における SSL の設定準備 489
- CA の認証の使用: 技術上の考慮事項 494
- SSL サーバとしての Integration Server の設定 495
- SSL クライアントとしての Integration Server の設定 497
- Integration Server JVM の SSL 情報の設定 498
- SSL サーバで複数のクライアント認証を使用する 500
- WS セキュリティと Integration Server 502

Integration Server における SSL の概要

管理者は、SSL (Secure Sockets Layer)/TLS (Transport Layer Security) を使用して Integration Server との送信/受信のセキュリティを確保するように Integration Server を設定することができます。この章では、Integration Server で SSL を使用する方法および Integration Server の SSL 認証の設定に必要な情報について説明します。

Integration Server の SSL 接続は、SSL サーバおよび SSL クライアント という観点から概要を把握すると、理解しやすくなります。SSL の接続要求を送信するのはクライアントです。次のいずれかがクライアントになることができます。

- パートナアプリケーションまたはリソース
- インターネットリソース
- Web ブラウザ
- Integration Server

SSL ハンドシェーク処理中、SSL サーバとして機能するエンティティは、要求元クライアントに X.509 形式の公開鍵認証 (デジタル認証とも呼ばれます) を提示して接続の要求に応答します。クライアントがこれらのクレデンシャルを認証した場合は、次のいずれかの動作が行われます。

- SSL 接続が確立されます。クライアントとサーバ間で情報を交換できます。

  または

- 認証プロセスの次のフェーズが発生します。サーバがクライアントの SSL クレデンシャルを要求します。サーバがこのクレデンシャル (クライアントの ID) を検証すると SSL 接続が確立され、情報の交換が行われます。

SSL ハンドシェーク中に、SSL クライアントと SSL サーバは、使用する SSL/TLS プロトコルバージョンおよび使用する暗号スイートのネゴシエーションを行います。クライアントとサーバが互いの公開認証を検証できないか、プロトコルバージョンまたは暗号スイートのネゴシエーションを行えない場合、SSL ハンドシェークは失敗します。SSL 接続は確立されません。

別の Integration Server またはパートナアプリケーションなどのクライアントが HTTPS または FTPS で Integration Server に要求をサブミットし、SSL 接続が確立された場合、Integration Server が SSL サーバ、クライアントが SSL クライアントとして機能します。

Integration Server サービスがインターネットリソースに HTTPS 要求または FTPS 要求をサブミットする場合、Integration Server が SSL クライアント、通信先のインターネットリソースが SSL サーバとして動作します。

サポートされている SSL 認証タイプ

Integration Server は一方向および双方向の SSL 認証タイプをサポートします。両方の認証タイプで、SSL サーバは認証を送信します。双方向認証で、SSL ハンドシェーク中に、SSL サーバは SSL クライアントからクライアント認証を要求します。

- 一方向SSL 認証では、安全なトランザクションの設定に備えて匿名クライアントがサーバのクレデンシャルを認証します。ほとんどの場合、クライアントのクレデンシャルを検証する必要はないため、サーバはクライアントのIDをチェックしません。ただし、必要に応じて、基本的なユーザ名/パスワードなどを使用してクライアントを認証することができます。

  このタイプの認証の代表としては、ブラウザがインターネットサーバに接続を確立して、安全なトランザクション (預金口座の表示やクレジットカードによる商品の購入など) を実行する接続があります。クライアントはサーバのクレデンシャルを認証してからトランザクションを開始する必要がありますが、サーバがすべてのクライアント (ブラウザ) を認証して、記録を残しておく必要はありません (また現実的ではありません)。

  Integration Server の場合、通常は、この接続タイプでパートナアプリケーションまたはリソースが Integration Server の真偽を検証する必要がありますが、パートナアプリケーションまたはリソース自体を認証する必要はありません。

- 双方向SSL 認証では、クライアントおよびサーバの両方が互いのクレデンシャルを認証してから、SSL 接続を確立して情報を交換する必要があります。

  一方向SSL 認証とは異なり、Integration Server と、パートナアプリケーションまたはリソースは、互いの認証、SSL 接続の確立および情報の送信を行うために、互いの SSL 認証にアクセスできる必要があります。双方向SSL 接続のセキュリティレベルは、一方向接続と比較してはるかに高くなります。

サポートされている SSL/TLS プロトコル

Integration Server JVM および無効にされた SSL/TLS プロトコルのリストによって、SSL/TLS のどのプロトコルが Integration Server との受信/送信接続にサポートされているかが決定されます。

Integration Server は以下の手順に従って、有効にされたプロトコルを決定します。

1. JVM によってサポートされている SSL/TLS プロトコルを取得する。

   受信接続については、Integration Server はポートの起動時にサポートされているプロトコルを取得します。送信接続については、Integration Server は送信接続の作成時にサポートされているプロトコルを取得します。

   JVM が異なると、サポートするプロトコルも異なります。したがって、Integration Server が SSL クライアントまたは SSL サーバとして機能する場合、Integration Server によって使用される JVM によって、使用できるプロトコルは影響を受けます。

   注:
   Integration Server は、JVM の java.security ファイルの上書きとして /config/security/webm_override_java.security を登録します。Integration Server はこのファイルを使用して、java.security ファイルにより制御される設定を上書きします。たと

えば、Java 11 用の java.security ファイルは TLSv1.1 および TLSv1 プロトコルを無効にします。これらのプロトコルを使用するかどうか選択するオプションをユーザが使用できるようにするため、Integration Server では、JVM の起動時に webm_override_java.security を適用して、無効になっているプロトコルリストを上書きします。TLSv1.1 および TLSv1 プロトコルを無効にするには、以下で説明するサーバ設定パラメータを使用できます。

2. 無効にされたプロトコルのリストを適用する。

   - watt.net.jsse.server.disabledProtocols サーバ設定パラメータは、ポートの受信接続に対して無効にされたプロトコルのリストを指定します。これらは、Integration Server が SSL サーバとして機能する場合に適用されます。
   - watt.net.jsse.client.disabledProtocols サーバ設定パラメータは、送信接続に対して無効にされたプロトコルのリストを指定します。これらは、Integration Server が SSL クライアントとして機能する場合に適用されます。

要するに、Integration Server は、JVM によってサポートされているすべてのプロトコル (webm_override_java.security ファイルによって上書きされていない java.security ファイルによって無効にされたプロトコルを除く)、および watt.net.jsse.server.disabledProtocols と watt.net.jsse.client.disabledProtocols の一方または両方のパラメータを自動的に有効にします。

例として、Azul Java 11 で利用できる JDK を使用する Integration Server 10.11 を考えます。付属する JVM は以下の SSL/TLS プロトコルをサポートします。

```
TLSv1.3 TLSv1.2 TLSv1.1 TLSv1 SSLv3 SSLv2Hello
```

Integration Server が SSL サーバとして機能する場合、watt.net.jsse.server.disabledProtocols パラメータのデフォルト値は次のプロトコルを無効にします: `SSLv2Hello`、`SSLv3`、`TLSv1`、`TLSv1.1`。

その結果、上記の Integration Server に対して有効にされた受信プロトコルは次のとおりです: `TLSv1.3 TLSv1.2`。

Integration Server が SSL クライアントとして機能する場合、watt.net.jsse.client.disabledProtocols パラメータのデフォルト値は次のプロトコルを無効にします: `SSLv2Hello`、`SSLv3`、`TLSv1`、`TLSv1.1`。

その結果、上記の Integration Server に対して有効にされた送信プロトコルは次のとおりです: `TLSv1.3 TLSv1.2`。

SSL サーバと SSL クライアントの両方またはいずれか一方として機能する Integration Server に許可される SSL/TLS プロトコルを変更するには、watt.net.jsse.server.disabledProtocols、watt.net.jsse.client.disabledProtocols のパラメータの値をそれぞれ変更します。

注:
ポートごとに JSSE 用のプロトコルを無効にすることができます。これは、HTTPS ポートと FTPS ポートにのみ適用されます。ポートごとに無効にされたプロトコルは、watt.net.jsse.server.disabledProtocols に指定されたプロトコルよりも優先されます。詳細については、231 ページの「ポートごとに JSSE 用のプロトコルを無効にする」を参照してください。

0006 サーバ SSL インタフェースログ機能が Debug レベルに設定されている場合、ポートが起動されるか送信接続が確立されると、Integration Server はサポートされているプロトコルに関するサーバログメッセージを書き込みます。たとえば、Integration Server がポートを起動するご

とに、Integration Server はそのポートにサポートされているプロトコルに関するサーバログメッセージを書き込みます。例: `[ISC.0006.0037D] (tid=344) SSL ポート 5543 はプロトコル {TLSv1.3, TLSv1.2} をサポートするために設定されています。`

送信接続が確立された場合、Integration Server はサポートされているプロトコルに関する同様のメッセージをログに記録します。例: `[ISC.0006.0038D] (tid=92) 送信 SSL 接続 www.softwareag.com:443 がプロトコル {TLSv1.3, TLSv1.2} をサポートするために設定されています。`

デフォルトで無効にされるプロトコルの watt.net.jsse* パラメータの変更

Integration Server 10.11 は、JSSE を使用するセキュアな受信/送信接続に対して TLSv1.3 をサポートし、このプロトコルはデフォルトで有効です。Integration Server 10.11 はデフォルトで有効なプロトコルとして TLSv1.1 を含みません。バージョン 10.7 より、Integration Server にはデフォルトの有効なプロトコルとして TLSv1.0 が含まれなくなりました。

デフォルトで有効なプロトコルでのこの変更をサポートするため、バージョン 10.7 より、Integration Server はサーバ設定パラメータ watt.net.jsse.client.enabledProtocols と watt.net.jsse.server.enabledProtocols を削除し、それぞれを watt.net.jsse.client.disabledProtocols と watt.net.jsse.server.disabledProtocols で置き換えます。

Integration Server の以前のバージョンからバージョン 10.7 以降への移行時に、Integration Server は watt.net.jsse.client.enabledProtocols と watt.net.jsse.server.enabledProtocols の現在の値を検出します。次に、Integration Server はこれらの値を使用して、watt.net.jsse.client.disabledProtocols と watt.net.jsse.server.disabledProtocols の値をビルドします。プロトコル SSLv2Hello、SSLv3、TLSv1、および TLSv1.1 が移行される watt.net.jsse.*.enabledProtocols リストにある場合でも、Integration Server はこれらのプロトコルを無効にされたプロトコルリストに自動的に追加します。

TLS の再ネゴシエーションの無効化

TLS の再ネゴシエーションを行うと、サービス拒否 (DoS) 攻撃を受ける可能性があります。Java のシステムプロパティを設定することで、JSSE を使用する HTTPS および FTPS のすべてのポートの TLS 再ネゴシエーションを無効にできます。設定するプロパティは、Integration Server で使用される JDK の JSSE プロバイダによって異なります。

- Oracle (SunJSSE) の JSSE プロバイダを使用する場合、TLS の再ネゴシエーションを無効にするには、Java システムプロパティ `jdk.tls.rejectClientInitiatedRenegotiation` を「true」に設定します。
- IBM (IBM JSSE2) の JSSE プロバイダを使用する場合、TLS の再ネゴシエーションを無効にするには、Java システムプロパティ `com.ibm.jsse2.renegotiate` を使用します。

注:
JSSE を使用しない HTTPS ポートおよび FTPS ポート (**Use JSSE** プロパティが **No** に設定) は iSaSiLk ライブラリを使用する Entrust for SSL を使用します。Integration Server には、再ネゴシエーションの無効化をサポートしない iSaSiLk Version 3.03 が付属しています。

SSL と使用する暗号スイートの指定

Integration Server で提供されるサーバ設定パラメータを使用して、受信/送信 SSL要求で使用できる暗号スイートを制御します。

次の表は、受信/送信 SSL接続に対する暗号スイートの使用を制御するサーバ設定パラメータを示します。

サーバ設定パラメータ	説明
watt.net.jsse.client.enabledCipherSuiteList	JSSE が有効であるか使用されている場合の送信 SSL 接続の暗号スイートを指定します。
watt.net.jsse.server.enabledCipherSuiteList	JSSE が有効であるか使用されている場合の受信 SSL 接続の暗号スイートを指定します。
watt.net.jsse.server.useCipherSuitesOrder	「true」に設定した場合、Integration Server は SSL/TLS ハンドシェーク中に watt.net.jsse.server.enabledCipherSuiteList で設定された暗号スイートの順序を尊重することを指定します。
watt.net.ssl.client.cipherSuiteList	JSSE が使用されていない場合の送信 SSL 接続の暗号スイートを指定します。
watt.net.ssl.server.cipherSuiteList	JSSE が使用されていない場合の受信 SSL 接続の暗号スイートを指定します。

許可された暗号スイートのサーバ設定パラメータではカンマ区切りリストを使用して許可された暗号スイートを識別しますが、パラメータの値としてファイルを使用することもできます。ファイルを使用すると、暗号スイートの長いリストを指定しやすくなります。

ファイルを使用して許可された暗号スイートを指定するときには、以下の情報に留意してください。

- ファイルでは、各暗号スイートを異なる行に指定します。
- 暗号スイートのリストの代わりにファイルを指定する対象の暗号スイートサーバ設定プロパティごとに、プロパティの値として以下の値を指定します。

  ```
  file:directoryName\filename
  ```

  例: `watt.net.jsse.server.enabledCipherSuiteList=file:c:\ssl\ciphers.txt`

- Integration Server は起動時に、ファイルおよびサポートしている暗号スイートのリストをロードします。Integration Server の起動後に行ったファイル内容の変更は、次回 Integration Server が起動するまで有効になりません。
- 暗号スイートサーバ設定パラメータの値はカンマ区切りリスト、デフォルト、またはファイルへの絶対パスに設定できます。上記の組み合わせを 1 つのパラメータに指定することはできません。

注:
ログ機能 0006 サーバ SSL インタフェースが Trace ログレベルに設定されている場合、受信ポートまたは送信ポートが起動されると、常に Integration Server は有効な暗号スイートに関するメッセージを書き込みます。

Integration Server における SSL の設定準備

Integration Server を SSL サーバとして設定するか SSL クライアントとして設定するかに関係なく、SSL に対する Integration Server 設定はいくつかの共通タスクから構成されます。主に、これらの共通タスクは SSL に必要な認証の準備に関係しています。

SSL 設定に対して Integration Server を準備するには、以下の上位レベルのタスクを完了してください。

1. **公開鍵/秘密鍵と認証を作成します**。これは一方向および双方向の SSL 認証に必要です。以下のアクティビティが含まれます。

   - 公開鍵/秘密鍵ペアの生成
   - CSR (Certificate Signing Request : 認証署名要求) の生成および CA (Certificate Authority : 認証局) への送信と署名
   - CA からの妥当性検査が行われた認証の受信
   - 署名付き認証のキーストアへのインポート

   公開鍵/秘密鍵と認証の作成の詳細については、Java ***keytool*** またはご使用の認証管理ツールのマニュアルを参照してください。

2. **キーストアとトラストストアを作成します**。これは一方向および双方向の SSL 認証に必要です。以下のアクティビティが含まれます。

   - キーストアの作成および署名付き認証と秘密鍵のインポート
   - トラストストアの作成および署名した CA の認証のインポート
   - 安全な IS 認証ディレクトリへのキーストアおよびトラストストアの格納
   - キーストアおよびトラストストアのエイリアスの作成詳細については、510ページの「キーストアの作成」を参照してください。

     重要:
     Oracle の keytool を使用してキーストアを作成する場合は、既存の秘密鍵をインポートできません。OpenSSL または Portecle などの他のツールを使用してください。

   キーストアとトラストストアの作成の詳細については、ご使用の認証管理ツールのマニュアルを参照してください。

3. **パートナアプリケーションまたはリソースの認証を取得し、必要に応じて、認証マッピングを作成します**。認証の取得は一方向および双方向の SSL 接続に必要です。認証マッピングの作成は双方向の SSL 認証に対して、また、Integration Server が SSL サーバとして機能している場合に必要です。

Integration Server Administrator を使用して以下の値を保存するアクティビティが含まれます。

- パートナの SSL 認証用 CA の署名付き認証
- パートナアプリケーションの署名付き認証 (SSL サーバとして機能する双方向の SSL 接続用)

認証のインポートおよび認証のマッピングの詳細については、553 ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。

4. **Integration Server の SSL 認証とキーを指定します**。これは一方向および双方向の SSL 認証に必要です。

   Integration Server Administrator を使用して以下の値を設定するアクティビティが含まれます。

   - Integration Server の識別に使用される SSL キー
   - 送信ドキュメント、メッセージ、およびデータストリームの署名に使用される秘密鍵
   - 受信ドキュメント、メッセージ、およびデータストリームの復号化に使用される秘密鍵

   Integration Server の SSL キーの設定の詳細については、次を参照してください: 492 ページの「Integration Server の SSL 認証とキーの指定」

SSL に必要な認証とキーを準備した後、SSL サーバと SSL クライアントの両方またはいずれか一方としての Integration Server の設定を実行できます。

- SSL サーバとしての Integration Server の設定には、ポートの作成、無効にされたプロトコルの指定、および許可された暗号スイートの指定が含まれます。詳細については、495 ページの「SSL サーバとしての Integration Server の設定」を参照してください。
- SSL クライアントとしての Integration Server の設定には、無効にされたプロトコルの指定および許可された暗号スイートの設定が含まれます。詳細については、497 ページの「SSL クライアントとしての Integration Server の設定」を参照してください。

Integration Server の公開鍵/秘密鍵と認証の作成

Integration Server を SSL サーバおよび SSL クライアントとして使用するには、公開鍵/秘密鍵のペアと認証が必要です。Integration Server の公開鍵/秘密鍵のペアの生成は、以下の一般的な手順から構成されます。この手順は使用される CA によって異なることがあります。

1. Integration Server の秘密鍵/公開鍵のペアを生成するには、標準の認証管理ツール (OpenSSL または Portecle など) を使用します。
2. 公開鍵は CSR に配置します。
3. CSR の作成後は CA に送信して、CSR を署名します。認証は DER 形式で要求します。PEM 形式 (または DER 以外の形式) で認証を受信した場合は、DER 形式に変換する必要があります。

署名した CA を認証することにより、Integration Server のデジタル認証に署名した CA の正統性が証明されます。Integration Server のデジタル認証が送信されるときに、CA からこの認証も送信されます。

4. CA から署名付き認証を受信した後、その認証をキーストアにインポートする必要があります。

これで、Integration Server で使用する SSL 認証および秘密鍵が設定されます。

通常は、約 1 年ごとまたは 2 年ごとに認証の更新が必要となったとき、鍵ペアを作成した後にこの手順を繰り返します。

Integration Server で妥当性検査する認証拡張が認証に含まれている場合は、watt.security.cert.wmChainVerifier.enforceExtensionsChecks サーバ設定プロパティを「true」に設定します。

キーストアとトラストストアの作成

キーストアおよびトラストストアは、SSL 認証、暗号化/復号化およびデジタル署名/検証サービスに必要な鍵と認証を格納するリポジトリとして機能するファイルです。個別のファイルで鍵と認証を管理する場合に比べ、キーストアおよびトラストストアを使用するとセキュリティが高くなり、管理もしやすくなります。

認証管理ツールを使用して、以下の手順に従います。

- SSL および対応するキーエイリアスに使用する Integration Server 鍵ペアを含むキーストアを、JKS 形式または PKCS12 形式で少なくとも 1 つ作成する
- サーバの鍵のペアの署名した CA (および必要に応じて認証チェーン) の信用のあるルート認証を含むトラストストアを、JKS 形式で少なくとも 1 つ作成する
- キーストアおよびトラストストアエイリアスを作成する

  キーストアおよびトラストストアを使った Integration Server の使用およびこれらのファイルのエイリアスを作成する方法については、509 ページの「Integration Server におけるキーストアとトラストストアの使用」を参照してください。

キーストアとトラストストアの作成、キーストアとトラストストアへの鍵と認証のインポートおよびこれらのファイルに関するその他の操作については、認証管理ツールのマニュアルを参照してください。

パートナアプリケーションまたはリソースの公開鍵と認証の取得

Integration Server が SSL サーバとして機能するか SSL クライアントとして機能するかに関係なく、パートナアプリケーションまたはリソースの認証を取得する必要があります。

Integration Server が、インターネット上の SSL で保護されたリソースを要求するサービスを実行する場合、Integration Server は SSL クライアントとして機能し、これらのリソースから認証を受信します。これらのトランザクションが機能するためには、Integration Server 上に他のリソースの公開鍵と署名した CA の認証のコピーが必要です。

Integration Server が一方向および双方向の認証の SSL サーバである場合、Integration Server は、SSL クライアントとして機能するリソースまたはパートナの SSL キーを署名した CA 認証を必要とします。これは、SSL ハンドシェーク処理中に行われる信頼検証に必要です。同様に、Integration Server は認証を送信し、クライアントはその認証を使用して信頼妥当性検査を実行します。双方向認証の場合、Integration Server にはクライアントの公開鍵認証も必要です。さらに、クライアント認証は Integration Server のユーザアカウントにマッピングする必要があります。

クライアント認証の Integration Server へのインポートおよびクライアント認証の設定の詳細については、549 ページの「クライアントの認証」を参照してください。ユーザへの認証のマッピングが必要となるのは、Integration Server が SSL サーバとして機能し、双方向認証が有効になっている場合のみであることに注意してください。

Integration Server の SSL キーと認証の設定

Integration Server が SSL を使用するようにする設定の一部として、サーバのデフォルト SSL 値を指定する必要があります。これらのデフォルト SSL 値は Integration Server をパートナに示し、送信の署名および受信の復号化に使用されます。

注:
ここで説明する SSL 認証設定は、Integration Server Web サービスおよび Integration Server XML ドキュメントの保護に使用する組み込みサービスで使用されます。詳細については、『*Web Services Developer's Guide*』および『*webMethods Integration Server Built-In Services Reference*』を参照してください。

Integration Server の SSL 認証を設定する前に、認証管理ツールを使用してキーストアとトラストストアを作成し、これらのキーストアとトラストストアのエイリアスを作成しておく必要があります。詳細については、491 ページの「キーストアとトラストストアの作成」を参照してください。

Integration Server の SSL 認証とキーの指定

Integration Server SSL 設定は、いくつかのグループに分かれています。次の設定グループには、[**キーストアエイリアス**] および [**キーエイリアス**] を選択できます。

- SSL キー。ここで指定する Integration Server の秘密鍵と公開鍵のペアは、要求元のパートナアプリケーション、インターネットリソースまたは Web サービスに Integration Server の SSL クレデンシャルを提示する場合に使用します。この設定により、Integration Server の SSL ID が決定されます。
- 署名キー。Integration Server からの送信ドキュメント、メッセージおよびデータストリームを署名する秘密鍵を指定します。
- 復号鍵。関連する Integration Server 公開鍵で情報が暗号化されている外部ソースからの受信ドキュメント、メッセージおよびデータストリームの復号化に使用する秘密鍵を指定します。

SSL 認証で署名した CA 認証の場所を指定するトラストストアの場合は、[**トラストストアエイリアス**] を指定します。一般に、これはデフォルト送信トラストストアと呼ばれます。

重要:
[**セキュリティ**] > [認証] ページの設定は、Integration Server の識別および任意の Integration Server ドキュメント、Web サービスまたは組み込みサービスで使用する SSL キーの指定に使用される、デフォルトの SSL 値です。さらに、Integration Server で作成される HTTPS ポートまたは FTPS ポートでは、そのポートにキーストア、キーエイリアス、トラストストアのすべてまたはそのいずれかが設定されていない場合、デフォルトのサーバ SSL キーおよびトラストストアエイリアスを使用します。したがって、[**セキュリティ**] > [認証] ページの値を変更する場合は、システム管理者またはセキュリティ管理者に問い合わせてください。

SSL 接続のために Integration Server を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [認証] に移動します。
3. [**認証設定の編集**] をクリックします。
4. [**SSL キー**] の下で、以下の手順に従います。
   - [**キーストアエイリアス**] リストで、サーバ認証に使用される秘密鍵と認証が入ったキーストアのユーザ指定識別子を選択します。
   - [**キーエイリアス**] リストで、上記のキーストアエイリアスで指定されたキーストアにある秘密鍵のユーザ指定テキスト識別子を選択します。
5. [**署名キー**] の下で、以下の手順に従います。
   - [**キーストアエイリアス**] リストで、送信メッセージの署名に使用される秘密鍵と認証が入ったキーストアのユーザ指定識別子を選択します。
   - [**キーエイリアス**] リストで、送信メッセージの署名のために指定されたキーストアにある秘密鍵のユーザ指定テキスト識別子を選択します。
6. [**復号キー**] の下で、以下の手順に従います。
   - [**キーストアエイリアス**] リストで、受信メッセージの復号化に使用される秘密鍵と認証が入ったキーストアのユーザ指定識別子を選択します。
   - [**キーエイリアス**] リストで、受信メッセージの復号化のために指定されたキーストアにある秘密鍵のユーザ指定テキスト識別子を選択します。
7. [**トラストストア**] の下の [**トラストストアエイリアス**] リストで、信頼検証に必要な CA 認証を含むトラストストアエイリアスを選択します。
8. [**変更内容の保存**] をクリックします。

メモ:

- Integration Server が SSL クライアントとして機能する場合、Integration Server が信頼検証用に [**トラストストア**] に指定されたトラストストアを使用するのは、トラストストアが送信

呼び出しを行うサービスで指定されていないときだけです (たとえば、pub.client:http サービスの *trustStore* 入力)。

- [**トラストストア**] 設定を空白にすると、呼び出されたサービスが信頼検証用のトラストストアを指定しない送信呼び出しの場合、Integration Server は watt.security.cert.wmChainVerifier.trustByDefault パラメータを使用して、Integration Server が送信要求の SSL ハンドシェーク中に受信する認証を受け入れ信頼するかどうかを決定します。「true」 (デフォルト) に設定されている場合、Integration Server は送信要求の SSL ハンドシェーク中に受信する任意の認証を受け入れ信頼します。watt.security.cert.wmChainVerifier.trustByDefault=false の場合、*trustStore* パラメータが指定されているか、[**セキュリティ**] > [認証] ページで [**トラストストア**] が設定されていない限り、Integration Server は送信 SSL 要求の SSL ハンドシェーク中に受信する任意の認証を受け入れません。
- Integration Server は、受信要求の認証または S/MIME 署名の妥当性検査のために暗黙的に認証を信頼することはありません。これらのいずれかの機能を使用する場合、サーバが信頼する CA の認証を含むトラストストアエイリアスを指定する必要があります。

CA の認証の使用: 技術上の考慮事項

次に、CA の認証およびトラストストアの使用に関する技術上の考慮事項を示します。

期限切れ CA の認証の処理

場合によっては、認証チェーン内にある CA の認証が有効期限を過ぎていることがあります。Web ブラウザは認証の有効期限が切れたインターネットリソースに接続する場合でも、接続を承認することがあります。CA の有効な認証にアクセスできる場合は、その認証の期限が切れていても、Web ブラウザは接続を承認できます。ただし、Integration Server は、接続を明示的に設定していない限り、期限切れの署名した認証がチェーン内に含まれているリソースには接続できません。

場合によっては、1 つ以上の CA の認証が期限切れになっているときに、Integration Server で接続を承認するように設定することがあります。このような場合は、watt.security.ssl.ignoreExpiredChains サーバ設定プロパティの値を変更する必要があります。詳細については、146 ページの「拡張設定の使い方」を参照してください。設定を変更したら、必ずサーバを再起動してください。

注:
期限切れの認証を無視すると、認証の期限切れによって接続を拒否する場合よりも、セキュリティレベルは低くなります。

トラストストアの使用のカスタマイズ

通常はトラストストアを指定しますが、指定しない場合もあります。たとえば、送信要求についてはすべての CA を信用し、受信要求については異なるポートの特定の CA を信用する場合などがあります。

送信要求に対し、以下のように指定できます。

- pub.client:http サービスの *trustStore* 入力パラメータなど、送信呼び出しを行うサービスのトラストストア。
- [**セキュリティ**] > [認証] ページの [**トラストストア**] 設定のデフォルト送信トラストストア。
- 送信要求の SSL ハンドシェーク中に受信する認証を受け入れ信頼するため、常に信頼するグローバル設定。watt.security.cert.wmChainVerifier.trustByDefault サーバ設定パラメータの値は、Integration Server が送信要求の SSL ハンドシェーク中に受信する認証を受け入れ信頼するかどうかを決定します。「true」に設定されている場合 (デフォルト)、Integration Server は受信する任意の認証を受け入れ信頼します。「false」に設定されている場合、Integration Server は送信 SSL 要求の SSL ハンドシェーク中に受信する任意の認証を受け入れません。

送信 SSL 要求のハンドシェーク中、Integration Server は最初にサービスに設定されているトラストストアをチェックします。指定されている場合、Integration Server は指定されたトラストストアを使用して、信頼検証を実行します。サービスがトラストストアを指定していない場合、Integration Server は要求の信頼検証用に [**セキュリティ**] > [認証] ページの [**トラストストア**] 設定に指定された値を使用します。[**セキュリティ**] > [認証] ページの [トラストストア] 設定が空白である場合、Integration Server は watt.security.cert.wmChainVerifier.trustByDefault の値を使用して、Integration Server が送信要求の SSL ハンドシェーク中に受信する任意の認証を受け入れ信頼するかどうかを決定します。

受信要求については、([**セキュリティ**] > [認証] ページを使用して) Integration Server レベルで、あるいはポートレベル ([**HTTPS ポート設定の編集**] ページまたは [**FTPS ポート設定の編集**] ページを使用) でトラストストアを指定できます。Integration Server レベルおよびポートレベルでトラストストアを指定しなかった場合、Integration Server は認証局を信頼しません。Integration Server レベルおよびポートレベルでトラストストアを指定すると、サーバはポートレベルで指定されているトラストストアを使用して、そのポートに対する接続が信用できるかどうかを判断します。トラストストアをポートレベルだけで指定すると、サーバはそのポートに対する要求について、ポートレベルの設定を使用します。

SSL サーバとしての Integration Server の設定

489 ページの「Integration Server における SSL の設定準備」で説明した一般的な SSL 設定タスクに加え、Integration Server を SSL サーバとして設定するには、ポートの作成、有効な SSL/TLS プロトコルの指定、および許可された暗号スイートの指定を行う必要もあります。

Integration Server を SSL サーバとして設定するには、以下の SSL サーバ固有のタスクを完了してください。

1. **HTTPS または FTPS ポートを追加します**。HTTPS または FTPS ポートがまだ定義されていない場合、作成する必要があります。これは一方向および双方向の SSL 認証に必要です。

   サーバへのセキュア接続のみを許可する場合は、以下の作業を行います。

   - プライマリポートが HTTPS ポートを使用することの確認
   - HTTPS ポート以外のすべてのポートの削除

   HTTPS または FTPS ポートは、必要に応じてさらに追加できます。

   ポートの作成の詳細については、167 ページの「ポートの設定」を参照してください。

2. **受信用の SSL/TLS プロトコルを指定します**。SSL サーバとして機能する Integration Server との通信用に許可される SSL/TLS プロトコルを指定するには、watt.net.jsse.server.disabledProtocols サーバ設定パラメータで明示的に無効にされた SSL/TLS プロトコルを実際に識別します。

   Integration Server がどのように SSL/TLS プロトコルの無効にされたリストを使用して許可された SSL/TLS プロトコルを識別するかの詳細については、485 ページの「サポートされている SSL/TLS プロトコル」を参照してください。

   ポートごとに JSSE 用の SSL/TLS プロトコルを無効にすることができます。ポートごとに無効にされたプロトコルは、watt.net.jsse.server.disabledProtocols に指定されたプロトコルよりも優先されます。ポートごとのプロトコルの無効化の詳細については、次を参照してください: 231 ページの「ポートごとに JSSE 用のプロトコルを無効にする」

   Java のシステムプロパティを設定することで、JSSE を使用する HTTPS および FTPS のすべてのポートの TLS 再ネゴシエーションを無効にできます。TLS の再ネゴシエーションを行うと、サービス拒否 (DoS) 攻撃を受ける可能性があります。TLS の再ネゴシエーションの無効化の詳細については、487 ページの「TLS の再ネゴシエーションの無効化」を参照してください。

   注:
   Integration Server が受信を保護するための JSSE を使用していず、代わりに TLSv1.0 を使用している場合、watt.net.ssl.server.handshake.minVersion および watt.net.ssl.server.handshake.maxVersion サーバ設定パラメータの値が有効なプロトコルを決定します。TLSv1.0 はセキュアではないことに注意してください。

3. **受信に対して許可された暗号スイートを指定します**。ポートが JSSE を使用して接続を保護する場合、watt.net.jsse.server.enabledCipherSuiteList は暗号スイートまたは受信 SSL 接続を指定します。

   オプションで watt.net.jsse.server.useCipherSuitesOrder=true を設定すると、SSL サーバとして機能する Integration Server は watt.net.jsse.server.enabledCipherSuiteList にある順序でその暗号スイートを提示します。必要に応じて、暗号スイートリストを並べ替え、必要とする強力な暗号スイートが最初に示されるようにしてください。

   有効な暗号スイートの識別の詳細については、488 ページの「SSL と使用する暗号スイートの指定」を参照してください。

   注:
   Integration Server が受信を保護するための JSSE を使用していず、代わりに TLSv1.0 を使用している場合、watt.net.ssl.server.cipherSuiteList および watt.net.ssl.client.strongcipheronly が許可された暗号スイートを決定します。

HTTPS または FTPS ポートの設定

Integration Server が SSL 接続を行うには、HTTPS または FTPS ポートが必要です。

サーバへのセキュア接続のみを許可する場合は、必ずプライマリポートが HTTPS または FTPS ポートを使用するように設定し、HTTPS と FTPS ポート以外のポートはすべて削除します。HTTPS または FTPS ポートは、必要に応じていくつでも追加できます。使用するポート上で受信待機しているアプリケーションが他にないことを確認します。

HTTPS、HTTPS 診断ポート、および FTPS ポートのすべてを設定して JSSE (Java Secure Socket Extension) を使用することで、さらに通信のセキュリティを確保できます。これには、ポートに対して [**JSSE を使用**] を [**はい**] に設定します。JSSE を使用するためのポートの設定の詳細については、167 ページの「ポートの設定」を参照してください。

注:
ポートが JSSE を使用するよう設定されていて、クライアントが提示する認証を Integration Server が信頼しない場合、Integration Server は要求を拒否します。Integration Server は基本認証 (ユーザ名/パスワード) を使用するようにはなりません。クライアントが有効な認証を提供したにもかかわらず、その認証にユーザがマップされていない場合、Integration Server は無効なクレデンシャルエラーを発行します。

サーバの SSL セキュリティレベルをポートごとに制御する方法

異なるポートに異なるサーバ認証を提示するように Integration Server を設定することができます。これによって、異なるポートから異なる SSL セキュリティレベルを提供することが可能になります。認証のセキュリティレベルは、認証の署名プロセスで決定します。必要な認証のクラスを認証局に知らせます。それに対し、認証局は該当する属性を持つ認証を作成します。その後ポートを設定するときに、ポートに関連付けるセキュリティレベルを持つ認証を指定します。

SSL クライアントとしての Integration Server の設定

489 ページの「Integration Server における SSL の設定準備」で説明した一般的な SSL 設定タスクに加え、SSL/TLS を SSL クライアントとして設定するには、ポートの作成、許可された SSL/TLS プロトコルの指定、および許可された暗号スイートの指定を行う必要があります。

Integration Server を SSL クライアントとして設定するには、以下の SSL クライアント固有のタスクを完了してください。

1. **送信用の SSL/TLS プロトコルを指定します。**SSL クライアントとして機能する Integration Server との通信用に許可された SSL/TLS プロトコルを指定するには、watt.net.jsse.client.disabledProtocols サーバ設定パラメータで明示的に無効にされた SSL/TLS プロトコルを実際に識別します。

   Integration Server がどのように SSL/TLS プロトコルの無効にされたリストを使用して許可された SSL/TLS プロトコルを識別するかの詳細については、485 ページの「サポートされている SSL/TLS プロトコル」を参照してください。

   注:
   Integration Server が送信を保護するための JSSE を使用していず、代わりに SSL/TLS の安全性が低いプロトコルである TLSv1.0 を使用している場合、watt.net.ssl.server.handshake.minVersion および watt.net.ssl.server.handshake.maxVersion サーバ設定パラメータの値が有効なプロトコルを決定します。TLSv1.0 はセキュアではありません。

2. **送信に対して許可された暗号スイートを指定します。**

   JSSE を使用して接続を保護する場合、watt.net.jsse.client.enabledCipherSuiteList は暗号スイートまたは送信 SSL 接続を指定します。有効な暗号スイートの識別の詳細については、488 ページの「SSL と使用する暗号スイートの指定」を参照してください。

注:
Integration Server が送信を保護するための JSSE を使用してず、代わりに TLSv1.0 を使用している場合、watt.net.ssl.client.cipherSuiteList および watt.net.ssl.client.strongcipheronly パラメータが許可された暗号スイートを決定します。

3. **送信接続が JSSE を使用することを確認します**。送信に JSSE を使用している場合 (強く推奨)、Integration Server API により開始された送信接続が JSSE を使用するように、以下のサーバ設定パラメータが「true」に設定されていることを確認してください。

   - watt.net.ssl.client.useJSSE はすべての送信 HTTPS 接続に対する JSSE の使用を制御します。
   - watt.net.ssl.client.ftps.useJSSE は送信 FTPS トラフィックに対する JSSE の使用を制御します。
   - watt.net.ssl.email.client.useJSSE は送信 SMTP 接続に対する JSSE の使用を制御します。

   注:
   個々のサービス呼び出しの設定は、上記のサーバ設定パラメータで制御されるグローバル設定を上書きできます。たとえば、pub.client:http サービスの *useJSSE* 入力パラメータは、その特定のサービス呼び出しの watt.net.ssl.client.useJSSE の値を上書きします。pub.client:http、pub.client:ftp、および他のサービスでの JSSE の使用の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

注:
Integration Server は、すべての SSL サーバに単一のクライアントの認証を提示できるだけでなく、さまざまな SSL サーバに対して異なるクライアント認証を提示することもできます。SSL サーバでの複数のクライアント認証の使用の詳細については、500 ページの「SSL サーバで複数のクライアント認証を使用する」を参照してください。

Integration Server JVM の SSL 情報の設定

データベース、LDAP などの Integration Server のサードパーティコンポーネントは、JSSE を使用して外部サーバへの送信呼び出しを行います。この場合、サードパーティコンポーネントは JSSE を直接使用して呼び出しを行います。サードパーティコンポーネントは Integration Server に対して設定されたデフォルトの SSL 設定を使用しません。代わりに、サードパーティコンポーネントが外部サーバと SSL を使用する場合、Integration Server JVM は SSL ハンドシェークの SSL コンテキストを作成するため、JVM の javax.net.ssl プロパティを使用します。

JVM 用の javax.net.ssl プロパティを使用して、キーストアの場所、トラストストアの場所、およびパスワード情報を設定する必要があります。ただし、これらのプロパティはストリング値をとり、結果的にファイルシステムのどこかにプレーンテキストでパスワード情報が保存されてしまいます。これは、セキュリティの脆弱性を示します。

このセキュリティの問題に対処するため、Integration Server では、パスワードがプレーンテキストで保存されない方法で、キーストアの場所、トラストストアの場所、および JVM のパスワードを指定できる方法を提供しています。具体的には、Integration Server Administrator は [**セキュリティ**] > [認証] ページに [**JVM キーストアエイリアス**] および [**JVM トラストストアエイリアス**] フィールドを用意しています。これらのフィールドを使用して、JVM でデフォルトの SSL コンテキストの設定に使用されるキーストアエイリアスおよびトラストストアエイリアスを

指定します。起動時に、Integration Server は、エイリアスからストアの場所とパスワードを取得することによって javax.net.ssl プロパティを設定します。

注:
[**JVM キーストアエイリアス**] および [**JVM トラストストアエイリアス**] フィールドの値はそれぞれ watt.server.ssl.keyStoreAlias および watt.server.ssl.trustStoreAlias サーバ設定パラメータに保存されます。

Integration Server JVM の SSL 情報を設定するには

1. Integration Server Administrator を使用して、デフォルト SSL コンテキストの作成に使用されるキーストアのキーストアエイリアスを作成します。

   キーストアエイリアスの作成の詳細については、513 ページの「キーストアエイリアスの作成」を参照してください。

2. DEFAULT_JVM_TRUSTSTORE トラストストアエイリアスを [**JVM トラストストアエイリアス**] として使用しない場合、Integration Server Administrator を使用して、SSL 対応サーバとの SSL 接続の確立に必要な情報が入ったトラストストアのトラストストアエイリアスを作成します。

   トラストストアエイリアスの作成の詳細については、515 ページの「トラストストアエイリアスの作成」を参照してください。

3. Integration Server Administrator を開き、[**セキュリティ**] > [認証] に移動します。

4. [**認証設定の編集**] をクリックします。

5. [JVM 設定] の下の [**JVM キーストアエイリアス**] リストで、手順 1 で作成したキーストアエイリアスを選択します。

6. トラストストアを使用する場合は、[**JVM トラストストアエイリアス**] リストで、手順 2 で作成したトラストストアエイリアスを選択します。

7. [**変更内容の保存**] をクリックします。

   保存時に、Integration Server は [**JVM キーストアエイリアス**] および [**JVM トラストストアエイリアス**] の値を使用して、watt.server.ssl.keyStoreAlias および watt.server.ssl.trustStoreAlias サーバ設定パラメータの値を更新します。

8. Integration Server を再起動します。Integration Server (および JVM) が再起動するまで、変更は有効になりません。

重要:
[**JVM キーストアエイリアス**] または [**JVM トラストストアエイリアス**] フィールドの値 (または関連するパラメータ watt.server.ssl.keyStoreAlias または watt.server.ssl.trustStoreAlias) の値を変更した場合、変更を有効にするには、Integration Server を再起動する必要があります。さらに、フィールドまたはパラメータによって参照されているキーストアとトラストストア

の両方またはそのいずれか一方の内容を変更した場合、変更を有効にするには、Integration Server を再起動する必要があります。

javax.net.ssl プロパティの優先順位

Integration Server では、JVM の javax.net.ssl プロパティをさまざまな方法で設定できます。たとえば、Integration Server が起動時にプロパティを設定するように custom_wrapper.conf ファイルを変更できます。また、Integration Server Administrator で [**セキュリティ**] > [認証] ページの [**JVM キーストアエイリアス**] および [**JVM トラストストアエイリアス**] フィールドを使用することもできます。さらに、サーバ設定パラメータとパッケージの起動時サービスを使用して、プロパティを設定できます。

Integration Server は次の優先順位を使用して、JVM の javax.net.ssl プロパティの値を設定します。

1. パッケージ起動時サービス
2. [**セキュリティ**] > [認証] の [**JVM キーストアエイリアス**] および [**JVM トラストストアエイリアス**] フィールド。これらのフィールドの値は、それぞれ watt.server.ssl.keyStoreAlias および watt.server.ssl.trustStoreAlias サーバ設定パラメータに保存されます。[**JVM キーストアエイリアス**] および[**JVM トラストストアエイリアス**] フィールドおよびサーバ設定パラメータの優先順位は同じレベルにあります。
3. watt.config.systemProperties サーバ設定パラメータ。
4. custom_wrapper.conf (Integration Server)

   Integration Server_directory /bin/setenv.bat の JAVA_CUSTOM_OPTS (Microservices Runtime)

重要:
javax.net.ssl プロパティの値に影響する変更を行った場合、変更を適用するには、Integration Server を再起動する必要があります。たとえば、以下のいずれかの操作を行った場合、Integration Server を再起動する必要があります。

- エイリアスが [**セキュリティ**] > [認証] ページの [**JVM キーストアエイリアス**] フィールドおよび watt.server.ssl.keyStoreAlias パラメータに設定されているキーストアを変更する。
- エイリアスが [**セキュリティ**] > [認証] ページの [**JVM トラストストアエイリアス**] フィールドおよび watt.server.ssl.trustStoreAlias パラメータに設定されているトラストストアを変更する。
- [**セキュリティ**] > [認証] ページの [**JVM キーストアエイリアス**] または [**JVM トラストストアエイリアス**] フィールドで使用されているエイリアスを変更する。
- watt.server.ssl.keyStoreAlias または watt.server.ssl.trustStoreAlias で使用されているエイリアスを変更する。
- custom_wrapper.conf または JAVA_CUSTOM_OPTS を編集して、javax.net.ssl パラメータの値を変更する。

SSL サーバで複数のクライアント認証を使用する

Integration Server は、すべての SSL サーバに単一のクライアントの認証を提示できるだけでなく、さまざまな SSL サーバに対して異なるクライアント認証を提示することもできます。さらに、Integration Server では他の組織によってこのために提供された認証を提示することもでき

ます(組織によっては、クライアントの認証を承認するのではなく、その組織の認証局が署名した認証をクライアントが使用することを希望する場合があります)。Integration Server が SSL サーバにどの認証を提示するかは、リモートサーバエイリアスまたは特別なパブリックサービスを使用して制御できます。

複数の組織からのクライアント認証を提示するように Integration Server を設定することもできます。この場合は、認証を取得して Integration Server 上に設定してから、リモートエイリアスまたは特殊なパブリックサービスを使用してどの認証を提示するかを制御します。

次の表では、さまざまなクライアント認証をさまざまな SSL サーバに提示するタスクのチェックリストを示します。

タスク	メモ
使用する各認証のコピーを取得する。	認証のキーストアを作成し、次に Integration Server を使用してキーストアエイリアスを作成します。認証の取得の詳細については、491 ページの「パートナアプリケーションまたはリソースの公開鍵と認証の取得」を参照してください。キーストアの作成の詳細については、491 ページの「キーストアとトラストストアの作成」を参照してください。キーストアエイリアスの作成の詳細については、513 ページの「キーストアエイリアスの作成」を参照してください。
リモートサーバエイリアスを設定する。	必須ではありませんが、特定の SSL サーバに特定の認証を送信するには、リモートサーバエイリアスを使用すると便利です。
異なる認証を提示するようにフローサービスをコーディングする。	フローサービスをコーディングする方法は、リモートサーバにリモートサーバエイリアスを定義したかどうかによって異なります。

リモートサーバエイリアスの設定

異なる SSL サーバに異なる認証を提示するには、リモートサーバエイリアスを使用すると便利です。

特定の認証を提示する SSL サーバにリモートサーバエイリアスを割り当てることから始めます(133 ページの「リモート Integration Server に対するエイリアスの設定」の手順を参照してください)。このエイリアスは、どの認証がリモートサーバに提示されるかを制御します。

フローサービスのコーディング

フローサービスをコーディングする方法は、通信を行う SSL サーバにリモートサーバエイリアスを定義したかどうかによって異なります。リモートサーバエイリアスを使用する場合は、どの認証を提示するかをエイリアスが制御します。リモートサーバエイリアスを定義すると、フローサービスに pub.remote:invoke サービスを使用して、リモートサーバ上のサービスを実行することができます。

リモートサーバエイリアスを定義していない場合は、Integration Server のパブリックサービスを使用して、認証の切り替えを行うためにフローサービスをコーディングする必要があります。パブリックサービスを使用して、Integration Server が提示するクライアント認証を制御できます。

次の表では、SSL サーバに提示されるクライアント認証を制御するために使用できるパブリックサービスを示します。

サービス	説明
pub.security.keystore:setKeyAndChain	提示する鍵および関連する認証チェーンを指定します。
pub.security:setKeyAndChainFromBytes	提示する鍵および関連する認証チェーンを指定します。鍵および認証情報は、ファイルではなく、バイト配列内にあります。
pub.security:clearKeyAndChain	一連のサービスで、鍵および認証チェーンをデフォルトに戻すために使用します。
pub.security:clearKeyAndChainFromBytes	一連のサービスで、鍵および認証チェーンをデフォルトに戻すために使用します。鍵および認証情報は、ファイルではなく、バイト配列内にあります。

上記のサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

WS セキュリティと Integration Server

Integration Server は、Web サービスで使用する WS セキュリティ標準をサポートします。HTTPS および FTPS などのトランスポートベースの認証フレームワークが、接続のエンドポイントを脅威から保護するのに対して、WS セキュリティはエンドポイント間のメッセージ伝送環境を保護します。SOAP メッセージヘッダーに格納される認証情報は、X.509 認証、ユーザ名トークンまたは SAML トークンに保存でき、実際の認証が含まれる場合または参照が含まれる場合があります。

この章では、Integration Server でトランスポートベースのセキュリティを使用することにフォーカスしています。ただし、Integration Server Web サービスを設定して WS セキュリティを使用する場合は、トランスポートベースのセキュリティを設定する場合と同様に、Integration Server Administrator で SSL キー、署名キーおよび復号キーを指定します (492 ページの「Integration Server の SSL 認証とキーの指定」を参照してください)。Integration Server ベースの Web サービスで WS セキュリティを使用する場合は、『*Web Services Developer's Guide*』および WS セキュリティに関する Oasis 標準のマニュアルを参照してください。

27 SSL セッションログの設定

- SSL セッションログの概要 504
- SSL セッションログの有効化 505
- SSL セッションログの表示 505
- SSL セッションログのデフォルト場所の変更 505
- SSL セッションログのサイズの管理 506
- ログの重複エントリを避けるには 506

SSL セッションログの概要

SSL セッションログには、受信接続用の SSL セッション情報が JSON 形式で含まれます。使用された暗号スイート、選択されたプロトコルバージョン、クライアントの詳細、およびサーバとセッションの作成の詳細などの情報が含まれます。この情報を使用して、正常な SSL ハンドシェイクの詳細を分析できます。

Integration Server は Entrust および JSSE セキュリティプロバイダ用の SSL セッション情報を取得します。以下に、JSSE および Entrust に対するサンプル SSL セッションログの一部を示します。

```
2019-07-15 11:21:49 IST {
  "provider" : "JSSE",
  "loggedInUser" : "Administrator",
  "sessionID" : "[93, 44, 20, 115, 150, 122, 228, 76, 181, 94, 62, 91, 207, 251,
   222, 105, 91, 14, 208, 203, 34, 115, 100, 112, 123, 71, 130, 212, 150, 225,
   7, 137]",
  "serverPort" : 12346,
  "creationTime" : 1563169907869,
  "lastAccessedTime" : 1563169909320,
  "selectedCipherSuite" : "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
  "selectedProtocol" : "TLSv1.2",
  "clientIPAddrees" : "127.0.0.1",
  "clientPort" : 64782
}
```

```
2019-07-15 11:21:34 IST {
  "provider" : "Entrust",
  "loggedInUser" : "Administrator",
  "sessionID" : "5D:FC:C7:77:19:ED:4B:B2:94:9A:DB:4D:51:9B:CD:96",
  "serverPort" : 12345,
  "creationTime" : 1563169893970,
  "lastAccessedTime" : 1563169894006,
  "selectedCipherSuite" : "TLS_RSA_WITH_AES_256_CBC_SHA",
  "selectedProtocol" : "TLSv1.0",
  "clientIPAddrees" : "127.0.0.1",
  "clientPort" : 64775
}
```

注:
これらのサンプルログは prettyPrint に対して設定されており、タイムスタンプが含まれています。prettyPrint とタイムスタンプの設定の詳細については、505 ページの「SSL セッションログの表示」を参照してください。

次の表に、SSL セッションログファイルにあるフィールドについて説明します。

フィールド	説明
`provider`	SSL セッションに使用されるセキュリティプロパティを識別します。Integration Server は SSL セッション用に JSSE および Entrust セキュリティプロバイダをサポートしています。
`loggedInUser`	SSL 接続を開始したクライアントのユーザ名を指定します。

フィールド	説明
sessionID	SSL ハンドシェイク時に作成されたセッション ID を指定します。
serverPort	要求を受信したポートを指定します。
creationTime	SSL セッションの作成時刻を指定します。
lastAccessedTime	SSL セッションが最後に使用された時刻を指定します。
selectedCipherSuite	SSL ハンドシェイクに使用された暗号スイートを指定します。
selectedProtocol	SSL ハンドシェイクに使用されたプロトコルバージョンを指定します。
clientIPAddrees	クライアントの IP アドレスを指定します。
clientPort	SSL 接続の確立に使用されたクライアントポート番号を指定します。

SSL セッションログの有効化

SSL セッションログを有効または無効にするため、Integration Server はサーバ設定パラメータを備えています。このパラメータの値を「true」に設定すると、Integration Server は SSL セッション情報を inboundSSLSessions.log に記録します。

watt.net.ssl.server.sessionlog サーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」のパラメータの説明を参照してください。

SSL セッションログの表示

SSL セッションログを表示するには、*Software AG_directory**Integration Server_directory*\instances*instance_name*\logs ディレクトリに移動して、テキストエディタで inboundSSLSessions.log ファイルを開きます。ただし、SSL セッションログファイルのデフォルトの場所は変更できます。

watt.net.ssl.server.sessionlog.prettyPrint パラメータを使用して、inboundSSLSessions.log ファイルを書式設定できます。このパラメータの値を「true」に設定すると、Integration Server は復帰 (CR) およびインデントを使用した JSON 書式を設定し、ログを読みやすくします。

SSL セッションログエントリにタイムスタンプを含めるには、watt.net.ssl.server.sessionlog.includeTimestamp パラメータの値を「true」に設定します。

watt.net.ssl.server.sessionlog.prettyPrint および watt.net.ssl.server.sessionlog.includeTimestamp サーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」のパラメータの説明を参照してください。

SSL セッションログのデフォルト場所の変更

ログデータを書き込む先のファイルの名前を含め、SSL セッションログのデフォルト場所を変更できます。watt.net.ssl.server.sessionlog.file サーバ設定パラメータはログファイルの場所と名前を制御します。

デフォルトで、SSL セッションログの場所は *Software AG_directory**Integration Server_directory*\instances*instance_name*\logs\inboundSSLSessions.log です。

SSL セッションログの場所またはファイルを変更するには、watt.net.ssl.server.sessionlog.file パラメータを変更して、Integration Server が SSL セッションログ情報を書き込む先のファイルへの絶対パスまたは相対パスを指定します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory*\instances*instance_name* に相対的なパスです。

パスにはディレクトリ名とファイル名を指定する必要があります。変更した内容を有効にするために、Integration Server を再起動します。

SSL セッションログのサイズの管理

デフォルトでは、Integration Server は SSL セッション情報を inboundSSLSessions.log に書き込みます。受信 HTTPS 要求の数が増加すると、inboundSSLSessions.log のサイズは急速に増加することがあります。大きな inboundSSLSessions.log ファイルは、リソースを消費するだけでなく、調査が難しくなります。この問題を避けるため、ファイルサイズに基づいて inboundSSLSessions.log ファイルを循環するように Integration Server を設定できます。

watt.net.ssl.server.sessionlog.maxFileSize サーバ設定パラメータを使用して、SSL セッションログのサイズ上限を指定できます。watt.net.ssl.server.sessionlog.maxFileSize が有効な値に設定されている場合、Integration Server は現在のログ名を変更して新しい inboundSSLSessions.log ファイルを作成することで、inboundSSLSessions.log ファイルを循環します。Integration Server はファイル名を inboundSSLSessions_<DATE(YYYYMMDD)>_TIME(HHMMSS).log に変更し、新しい inboundSSLSessions.log ファイルを作成します。

watt.net.ssl.server.sessionlog.maxFileSize サーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」のパラメータの説明を参照してください。

ログの重複エントリを避けるには

Integration Server を設定して、セッションログエントリをキャッシュすることにより SSL セッションログの重複エントリを避けることができます。watt.net.ssl.server.sessionlog.cacheLogEntries サーバ設定パラメータは、Integration Server に SSL セッションログエントリをキャッシュで追跡するように指示します。このパラメータの値を「true」に設定すると、Integration Server は SSL セッションエントリをキャッシュに書き込みます。セッションが再使用され、キャッシュに存在する場合、Integration Server はそのセッションの SSL セッション情報を再度ログに記録しません。これによって、ログファイルの重複エントリはなくなります。

ただし、SSL セッションエントリがキャッシュに保持される時間は、watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime サーバ設定パラメータを使用して設定されます。キャッシュ内の SSL セッションエントリは、watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime サーバ設定パラメータに設定された時間後に期限切れになります。Integration Server はそのキャッシュから期限切れの SSL セッションエントリを削除します。Integration Server がログエントリを削除したセッションをクライアントが再使用する場合、Integration Server はそのセッションのセッション情報を再度ログに記録します。

Integration Server は SSL セッションログエントリスイーパというスイーパタスクを使用して、期限切れセッションを削除します。

watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime サーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」のパラメータの説明を参照してください。

28 Integration Server におけるキーストアとトラストストアの使用

- キーストアとトラストストア 510
- Integration Server によるキーストアおよびトラストストアの使用方法 511
- キーストア、トラストストアおよび鍵のエイリアス 512

キーストアとトラストストア

Integration Server の秘密鍵および SSL 認証は、キーストアファイルに保存されます。認証の信用のあるルートはトラストストアファイルに保存されます。キーストアおよびトラストストアは、業界標準のファイル形式を持つセキュアなファイルです。

キーストアファイル

Integration Server は、「キーストア」と呼ばれる特殊なファイルを使用して、SSL 認証と鍵を保存します。

キーストアファイルには、秘密鍵とそれに対応する公開鍵の署名付き認証からなるペアが、1 つ以上含まれています。キーストアはパスワードを使用して厳格に保護し、管理者のみがアクセス可能な方法で (ファイルシステムまたはそのほかの場所に) 保存する必要があります。

キーストアファイルの形式

Integration Server のキーストアで使用するデフォルトの認証ファイル形式は、Oracle が提供する独自のキーストア実装方式の JKS (Java keystore) です。また、PKCS12 を使用することもできます。これは広く使用されている標準化された認証ファイル形式であり、高度な移植性を提供します。他のキーストアタイプを使用可能にするには、以下の作業を行います。

- 追加セキュリティプロバイダのロード
- watt.security.keyStore.supportedTypes プロパティの設定

HSM ベースのキーストア

Integration Server は、特定の条件下で、HSM (Hardware Security Module) に保存されたキーストアファイルの使用をサポートします。Integration Server はポートの HSM ベースのキーストアをサポートしていますが、他のコンポーネントのキーストアはサポートしていません。HSM ベースのキーストアを、リモートサーバエイリアス、送信認証、内部サーバポート、WS セキュリティまたは Integration Server パブリックサービスと共に使用することはできません。

現在サポートされているのは、nCipher ハードウェアカードモジュールのみです。

キーストアの作成

Oracle の Java 認証エディタである keytool を使用して、コマンドラインから Integration Server キーストアを作成および管理できます。

OpenSSL および Portecle など、他の標準ツールを使用することもできます。

注:
キーストアファイルを作成または管理するための、Software AG 独自のキーストアユーティリティセットは用意されていません。

トラストストアファイル

Integration Server は、「トラストストア」を使用して、署名した CA の公開鍵である信用のあるルート認証を保存します。トラストストアは認証チェーン全体の信用のあるルートを含むことができますが、Integration Server トラストストア内の認証組織に関して要件は一切ありません。単純に、指定した信用のあるディレクトリ内の CA のすべての公開鍵を含むデータベースとして機能します。

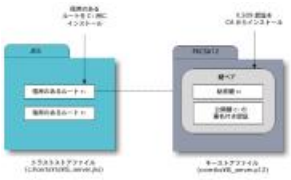
トラストストアファイルの形式

トラストストアのデフォルト形式は JKS です。JKS は、Oracle が提供する独自のキーストア実装方式です。Oracle の Java 認証エディタである keytool を使用して、コマンドラインから JKS トラストストアを作成および管理できます。

Integration Server によるキーストアおよびトラストストアの使用方法

Integration Server コンポーネントを SSL 認証するには、認可済みの有効な X.509 認証がキーストアファイルにインストールされ、認証の発行元 (CA) の秘密鍵および署名した認証がトラストストアファイルにインストールされている必要があります。次の図に、これらの要件および 2 つのファイルの関係を示します。

トラストストアファイルとキーストアファイルの関係を示す例



上の図に示すように、1 つのトラストストアファイルに複数の信用のあるルート認証 (署名した CA の公開鍵) を含めることができます。これらの信用のあるルートを、複数のキーストアファイルと関連付けることができます。1 つのキーストアファイルには、複数の Integration Server コンポーネントの鍵ペアおよびコンポーネントの認証に必要な認証チェーン全体を含めることができます。

認証チェーンでは、信用のある CA の認証に到達するまで、リスト内の後続の署名の妥当性検査を 1 つずつ行う必要があります。Integration Server の場合は、キーストアおよびトラストストアに認証チェーン全体を含める必要があります。また、Integration Server のトラストストアに、クライアントが使用するルート CA 認証が含まれている必要があります。

キーストアファイルとトラストストアファイルの保護

キーストアファイルおよびトラストストアファイルは、オペレーティングシステムのファイルシステム内に存在します。これらのファイルは非常に重要なため、セキュアなディレクトリパスに保存する必要があります。これらのファイルのいずれかが見つからない場合、Integration Server

認証は無効になり、サーバとの接続は行われません。これらの認証ファイルへのユーザアクセスは、My webMethods または Integration Server の管理者に限定することをお勧めします。

キーストア、トラストストアおよび鍵のエイリアス

特定のキーストアファイルまたはトラストストアファイル、あるいはキーストア内の秘密鍵を識別するため、Integration Server ではエイリアスを使用します。このエイリアスは、Java keytool および他の認証管理ツールを使用する場合と同じように使用できます。エイリアスを使用すると、Integration Server Administrator でキーストア、トラストストアまたは秘密鍵を指定するとき、あるいは Integration Server パブリックサービスを使用するときにパス情報を入力する必要がないため、キーストアおよびトラストストアの管理が単純化されます。

キーストアおよびトラストストアのエイリアスは、Integration Server Administrator の [**セキュリティ**] > [キーストア] パネルから表示、作成および編集できます。

注:
キーストアおよびトラストストアのエイリアスの大文字と小文字は区別されません。

キーエイリアスは、キーストア内にある特定の鍵のラベルです。キーエイリアスは、サードパーティの認証管理ツールを使用して作成します。Integration Server Administrator ではキーエイリアスを作成しませんが、Integration Server Administrator で適切なキーエイリアスを選択して、SSL 認証、署名および復号化に使用する鍵を指定する必要があります。

デフォルトのキーストアおよびトラストストアのエイリアス

Integration Server は、初めて Integration Server が起動するときに、デフォルトのキーストアおよびトラストストアのエイリアスを作成します。

重要:
これらのエイリアスは、テストおよびデバッグが実行される開発環境でのみ使用し、実稼働環境では使用しないでください。

以下の表に、Integration Server の事前定義済みのキーストアおよびトラストストアのエイリアスを示します。

デフォルトのエイリアス	説明
DEFAULT_IS_KEYSTORE	Integration Server のデフォルトのキーストアのエイリアスの名前。DEFAULT_IS_KEYSTORE は、*Software AG_directory* /common/conf/keystore.jks ファイルを指し示します。すべてのエントリでデフォルトのパスワードは「manage」です。
DEFAULT_IS_TRUSTSTORE	Integration Server のデフォルトのトラストストアのエイリアスの名前。DEFAULT_IS_TRUSTSTORE は、*Software AG_directory* /common/conf/platform_truststore.jks ファイルを指示します。すべてのエントリでデフォルトのパスワードは「manage」です。

デフォルトのエイリアス	説明
DEFAULT_JVM_TRUSTSTORE	JVM トラストストアのトラストストアエイリアスの名前。DEFAULT_JVM_TRUSTSTORE は {java.home}/lib/security/cacerts を指し示します。

デフォルトのエイリアスは編集および削除できます。デフォルトのキーストアおよびトラストストアの認証ファイル形式は、JKS (Java keystore) です。デフォルトのキーストアおよびトラストストアのエイリアスのプロバイダは、JVM に付属しています。

キーストアエイリアスの作成

Oracle Java keytool または別のサードパーティの認証ツールでキーストアファイルを作成した後、キーストアにエイリアスを割り当てます。

キーストアファイルのエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [キーストア] に移動します。
3. [**キーストアエイリアスの作成**] をクリックします。
4. [**キーストアのプロパティ**] 設定に次のように入力します。

設定項目	指定内容
[エイリアス]	キーストアファイルのテキスト識別子。 キーストアには、Integration Server、パートナアプリケーションまたは Integration Server コンポーネントの秘密鍵と認証 (関連する公開鍵を含む) が保存されます。
説明	オプション。キーストアエイリアスのテキスト説明。
[タイプ]	キーストアファイルの認証ファイル形式。キーストアの場合、デフォルトは JKS です。キーストアに PKCS12 形式を使用することもできます。 他のキーストアタイプを使用可能にするには、以下の作業を行います。 ■ 追加セキュリティプロバイダのロード ■ watt.security.keyStore.supportedTypes サーバ設定パラメータの設定

設定項目	指定内容
[プロバイダ]	キーストアまたはトラストストアのタイプに使用されるプロバイダ。デフォルトプロバイダは、JVM に組み込まれているプロバイダです。Oracle、IBM またはその他のプロバイダを指定できます。 **注:** キーストアタイプが PKCS12 である場合、プロバイダ「BC」は Bouncy Castle の省略形です。 通常、使用する HSM デバイスをデフォルトプロバイダがサポートしていない場合にのみ、プロバイダを指定する必要があります。 別のプロバイダを設定すると、デフォルト以外のキーストアタイプがサポートされます。Integration Server は、キーストアの場合は PKCS12 および JKS をどちらもサポートしますが、トラストストアの場合は JKS のみをサポートします。
[場所]	サーバ上にあるキーストアファイルの場所へのパス。 完全パス名または Integration Server に対する相対パスを指定できます。
[パスワード]/[パスワードの再入力]	このエイリアスに関連付けられている保存済みキーストアファイルのパスワード。 キーストアにパスワードが必要な場合は、キーストアの作成時にキーストアユーティリティを使用して定義しておく必要があります。キーストアエイリアスの作成が済むと、キーストアパスワードは Integration Server 送信パスワードとして自動的に保存されます。 対応するキーストアエイリアスを管理するときに、キーストアパスワードを必ず使用可能にしておいてください。キーストアにパスワードが不要な場合は、フィールドを空白のままにします。
[HSM ベースのキーストア]	キーストアファイルを HSM デバイス上に保存するかどうかを指定します。現在サポートされているのは、nCipher ハードウェアカードモジュールのみです。 このオプションを選択した場合は、**[場所]** フィールドにパスを指定しません。

5. **[サブミット]** をクリックします。

6. **[キーエイリアス]** 設定に次のように入力します。

設定項目	指定内容
[パスワード]/[パスワードの再入力]	キーストア内にある各エイリアスのパスワード。

設定項目	指定内容
	ほとんどのエイリアスにはパスワードが必要です。何らかの理由により、Integration Server がこのエイリアスを使用する場合は、そのエイリアスのパスワードを指定する必要があります。
[なし]	このエイリアスにはパスワードが必要ないことを示します。 パスワードで保護されていないキーストア内のエイリアスに、この設定を選択します。

7. [**変更内容の保存**] をクリックします。

トラストストアエイリアスの作成

トラストストアファイルにエイリアスを割り当てるには、以下の手順に従います。

トラストストアファイルのエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [キーストア] に移動します。
3. [**トラストストアエイリアスの作成**] をクリックします。
4. [**トラストストアのプロパティ**] 設定に次のように入力します。

設定項目	指定内容
[エイリアス]	トラストストアファイルのテキスト識別子。 トラストストアには、Integration Server、パートナアプリケーションまたは Integration Server コンポーネントの信用のある CA の認証が保存されます。
説明	オプション。トラストストアエイリアスのテキスト説明。
[タイプ]	トラストストアの認証ファイル形式。デフォルトは JKS です。 他のトラストストアタイプを使用可能にするには、以下の作業を行います。 ■ 追加セキュリティプロバイダのロード ■ watt.security.trustStore.supportedTypes サーバ設定プロパティの設定

設定項目	指定内容
[プロバイダ]	このトラストストアタイプに使用されるプロバイダ。デフォルトプロバイダは、JVM に組み込まれているプロバイダです。Oracle、IBM またはその他のプロバイダを指定できます。 使用する HSM デバイスをデフォルトプロバイダがサポートしていない場合にのみ、プロバイダを指定します。 別のプロバイダを設定すると、デフォルト (JKS) 以外のキーストアタイプがサポートされます。ただし、Software AG では、それらのキーストアタイプの使用をサポートしていません。
[場所]	サーバ上にあるトラストストアファイルの場所へのパス。 完全パス名または Integration Server に対する相対パスを指定できます。
[パスワード]/[パスワードの再入力]	トラストストアの内容を保護するために使用するパスワードの指定。 トラストストアの作成時に、キーストアユーティリティを使用してこのパスワードを定義しておく必要があります。トラストストアエイリアスの作成が済むと、そのパスワードは Integration Server 送信パスワードとして自動的に保存されます。 対応するトラストストアエイリアスを管理するときに、トラストストアパスワードを必ず使用可能にしておいてください。
[セカンダリトラストストア]	セカンダリトラストストアとして使用されるトラストストアのエイリアス。セカンダリトラストストアはプライマリトラストストア (つまり、このエイリアスの作成対象のトラストストア) の拡張として機能します。 たとえば、組織のすべてのプライベート CA 認証を含む「CompanyTruststore」という名前のトラストストアエイリアスを作成するとします。DEFAULT_JVM_TRUSTSTORE からの信用のある CA 認証を信頼検証に含める必要がある場合、DEFAULT_JVM_TRUSTSTORE をセカンダリトラストストアとして設定します。信頼検証を実行すると、Integration Server は最初に発行元の「CompanyTruststore」をチェックします。一致がない場合、Integration Server はセカンダリトラストストアをチェックします。この例では、セカンダリトラストストアを使用することで、プライベート認証をパブリック認証から区別して、かつ、両者を信頼認証に含めることができます。 **注:** DEFAULT_JVM_TRUSTSTORE は JVM トラストストアのトラストストアエイリアスです。

5. [**変更内容の保存**] をクリックします。

29 リソースへのアクセス制御

- 概要 518
- リソースへのアクセスをポートごとに制御 519
- ディレクティブの使用の制御 532
- ACL によるリソースへのアクセス制御 534
- ブラックリストへのサービスの追加 545

概要

サーバは、クライアントからサービスへのアクセス要求を受け取ると、複数のチェックを実行して、そのクライアントがサービスにアクセスする許可を得ているかどうかを確認します。サーバは次の順番でチェックを行います。サービスにアクセスするには、クライアントがすべてのチェック項目をクリアしている必要があります。

1. クライアントの IP アドレスからポートへの接続は許可されているか

   サーバは、このポート経由でサーバに接続することが許可されている IP アドレスの許可/拒否リストをチェックします。ポートが Enterprise Gateway 外部ポートであり、かつ、サーバが webMethods Enterprise Gateway に対してライセンスされている場合は、サーバでは Enterprise Gateway 拒否リストもチェックします。許可されている IP アドレスであれば、サーバは次のテストを実行します。許可されていない場合、サーバは要求を拒否します。

2. 要求されたサービスはこのポートからの使用が可能か

   サーバは、このポートから実行できるサービスの許可/拒否リストをチェックします。サービスがこのポートから実行できるものであれば、サーバは次のテストを実行します。実行できないものであれば、サーバは要求を拒否します。サーバは、サービスの実行要求に対してのみこのテストを行います。サービスのリスト、読み取りまたは書き込みアクセスに対しては、テストは行われません。

3. 要求されたサービスはブラックリストに記載されているか

   サーバは、すべてのユーザに対する呼び出しがブロックされているサービスを指定するサービスブラックリストをチェックします。サービスがブラックリストに記載されている場合、サーバはサービスへのアクセスを拒否し、要求を拒否します。サービスがブラックリストに記載されていない場合、サーバは次のテストを実行します。

4. 要求しているユーザは、このサービスへのアクセスを許可されているか

   サーバは要求に関連付けられているユーザ名を、サービスに関連付けられている適切な ACL (Access Control List : アクセスコントロールリスト) と照合します。

   サーバは、ユーザ名を、サービスに関連付けられているリスト ACL、読み取り ACL、書き込み ACL、実行 ACL と照合します。ユーザが ACL にリストされているグループに属する場合、サーバは要求を承認します。実行できないものであれば、サーバは要求を拒否します。

次の設定は Integration Server Administrator を使用して設定できます。

- ポートに接続する IP アドレスを制限する方法については、下記の519 ページの「ポートに接続可能な IP アドレスの制限」を参照してください。
- ポートから使用可能なサービスを制限する方法については、526 ページの「ポートから使用可能なサービスまたは Web サービス記述子の制限」を参照してください。
- ブラックリストにサービスを追加するには、次を参照してください:545 ページの「ブラックリストへのサービスの追加」
- エレメントにアクセスできるユーザを ACL を使用して制御する方法については、534 ページの「ACL によるリソースへのアクセス制御」を参照してください。

リソースへのアクセスをポートごとに制御

デフォルトでは、Integration Server は 5555 に HTTP ポートを設定します。このポートは IP アドレスによって識別されるすべてのホストが接続を許可されており、さらに ACL によって制限されない限り、そのポートを通じてすべてのサービスへのアクセスが許可されます。このポートは Integration Server の初期のインストールおよび設定時、また多くの開発環境においては最適ですが、実際の展開時にはこのポートをパートナやユーザなどの指定された IP アドレスのみ接続を許可し、指定されたサービスのみ使用可能にするようなポートに置き換える必要があります。

注:
またデフォルトでは Integration Server は診断ポートを 9999 に設定します。そのポートではすべてのホストがサーバへの接続を許可されます。ただし、ユーザがアクセスできるのは、Administrators ACL で定義されたサービスのみです。

ここでは、ポートレベルでリソースへのアクセスを制御する方法について説明します。ACL を使用してアクセスを制御する方法については、534 ページの「ACL によるリソースへのアクセス制御」を参照してください。

ポートに接続可能な IP アドレスの制限

任意のポートについて、IP アクセスを次の 2 つのうちいずれかに指定できます。

- [**デフォルトで拒否**]。明示的に許可したものを除くすべてのホストからの要求を*拒否*するようにポートを設定します。この方法は、大部分のホストを拒否して少数のホストを許可する場合に使用します。
- [**デフォルトで許可**]。明示的に拒否したものを除くすべてのホストからの要求を*許可*するようにポートを設定します。この方法は、大部分のホストを許可して少数のホストを拒否する場合に使用します。

これらの設定は、**グローバル** (すべてのポート) にまたは**個別** (1 つのポート) に指定できます。

注:
ポートが Enterprise Gateway 外部ポートであり、かつ、サーバが webMethods Enterprise Gateway に対してライセンスされている場合、IP アドレスを Enterprise Gateway ルールでさらに制限できます。IP アドレスからの要求がルールに違反する場合、サーバはその IP アドレスを Enterprise Gateway によって管理される拒否リストに追加します。

次の表は、さまざまなタイプの IP アクセスの割り当てについて説明の参照先を示しています。

アクセスのタイプ	説明の参照先
IP アクセスをグローバルに制御	
[デフォルトで拒否]	520 ページの「特定のホストからの受信接続を許可 (その他すべてを拒否)」
[デフォルトで許可]	521 ページの「特定のホストからの受信接続を拒否 (その他すべてを許可)」

アクセスのタイプ	説明の参照先
個別のポートの IP アクセスを制御	
[デフォルトで拒否]	522 ページの「特定のホストからの受信要求を許可 (その他すべてを拒否)」
[デフォルトで許可]	524 ページの「特定のホストからの受信要求を拒否 (その他すべてを許可)」

注:
Software AG Command Central では、ポートへの IP アクセスを制御する機能は IP アクセス制限と呼ばれます。

すべてのポートへの IP アクセスを制御 (グローバル)

ここでは、ポートに対してグローバルに IP アクセスを設定する方法について説明します。サーバはこの設定を使用して、カスタム IP アクセスが設定されていないポートへの IP アクセスを決定します。デフォルトのグローバル設定は [デフォルトで許可] です。

ポートを作成するときに、ポートへの IP アクセスをカスタマイズしたり、またはポートでサーバのグローバル IP アクセス設定を使用するように指定したりできます。グローバル IP アクセス設定を使用しており、後でその設定を変更した場合、サーバはポートに対して新しいグローバル設定を使用します。たとえば、サーバの出荷時に、グローバル IP アクセス設定として (明示的に拒否されたホストがない) [デフォルトで許可] が使用されていたとします。新しいポート 6666 を作成し、そのポートへの IP アクセスをカスタマイズしなければ、サーバはポート 6666 に対して [デフォルトで許可] を使用します。その後、グローバル IP アクセスを [デフォルトで拒否] に変更すると、サーバはポート 6666 に対して [デフォルトで拒否] を使用します。さらにその後、ポート 6666 への IP アクセスをカスタマイズすると、グローバル設定に対するその後の変更は、ポート 6666 には影響しません。

個別のポートへの IP アクセスをカスタマイズする方法については、522 ページの「特定のホストからの受信要求を許可 (その他すべてを拒否)」および 524 ページの「特定のホストからの受信要求を拒否 (その他すべてを許可)」を参照してください。

特定のホストからの受信接続を許可 (その他すべてを拒否)

以下の手順は、グローバル IP アクセス設定を [デフォルトで拒否] に変更して、アクセスを許可するホストを特定する方法について説明しています。

この設定を有効にすると、サーバは大部分のホストを拒否し、少数のホストを許可します。

重要:
グローバル設定を [デフォルトで拒否] に切り替える前に、グローバル設定に依存しないポートが少なくとも 1 つあり、そのポートで少なくとも 1 つのホストが許可されていることを確認してください。すべてのホストをサーバからロックアウトしてしまった場合は、適切な設定ファイルを手動で更新することで問題を解決できます。手順については、『525 ページの「誤ってすべてのホストに対して IP アクセスを拒否してしまった場合」』を参照してください。

特定のホストからの受信要求のみを許可するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**グローバル IP アクセス制限の変更**] をクリックします。
4. [**IP アクセスモードのデフォルトを拒否に変更**] をクリックします。

   アクセスモードが変更され、許可リストにホストを追加するページが表示されます。Integration Server Administrator を使用しているマシンのホスト名と IP アドレスが既に含まれ、サーバからロックアウトされないようになっていることが確認できます。
5. [**許可リストにホストを追加**] をクリックします。
6. サーバが受信要求を受け入れるホストのホスト名 (`workstation5.webmethods.com` など) または IP アドレス (`132.906.19.22` や `2001:db8:85a3:8d3:1319:8a2e:370:7348` など) を指定します。エントリは「`*.allowme.com, *.allowme2.com`」のようにカンマで区切ります。

   ホスト名または IP アドレスには大文字および小文字のアルファベット文字、数字 (0-9)、ハイフン (-) およびピリオド (.) を含めることができますが、スペースは含めることができません。IPv6 の場合、IP アドレスにコロン (:) およびカッコ ([]) も使用できます。

   ホストの完全修飾ドメイン名は使用しないでください。Integration Server は受信ホスト名を単純なホスト名に解決した後、単純なホスト名を許可リストの完全修飾ドメイン名に比較します。名前は一致せず、Integration Server では、要求を拒否すべきと判断します。これに回避するには、単純なホスト名の末尾にワイルドカード * を使用できます。または、IP アドレスを使用します。

   注:
   IP アドレスは送信偽造が困難なので、より高いセキュリティレベルが保証されます。

   次のようなパターンマッチング文字を使用すると、類似したホスト名や IP アドレスを持つ複数のクライアントを識別できます。

文字	説明	例
*	任意の数の文字に対応	r*.webmethods.com
?	任意の 1 文字に対応	workstation?.webmethods.com

7. [**ホストの追加**] をクリックします。

特定のホストからの受信接続を拒否 (その他すべてを許可)

以下の手順は、グローバル IP アクセス設定を [デフォルトで許可] に変更して、アクセスを拒否するホストを特定する方法について説明しています。

この設定を有効にすると、サーバは大部分のホストを許可し、少数のホストを拒否します。

特定のホストからの受信要求を拒否するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**グローバル IP アクセス制限の変更**] をクリックします。
4. [**IP アクセスモードのデフォルトを許可に変更**] をクリックします。

   アクセスモードが変更され、拒否リストにホストを追加するページが表示されます。
5. [**拒否リストにホストを追加**] をクリックします。
6. サーバが受信要求を拒否するホストのホスト名 (`workstation5.webmethods.com` など) または IP アドレス (`132.906.19.22` や `2001:db8:85a3:8d3:1319:8a2e:370:7348` など) を指定します。エントリは「`*.denyme.com, *.denyme2.com`」のようにカンマで区切ります。

   ホスト名または IP アドレスには大文字および小文字のアルファベット文字、数字 (0-9)、ハイフン (-) およびピリオド (.) を含めることができますが、スペースは含めることができません。IPv6 の場合、IP アドレスにコロン (:) およびカッコ ([]) も使用できます。

   ホストの完全修飾ドメイン名は使用しないでください。Integration Server は受信ホスト名を単純なホスト名に解決した後、単純なホスト名を拒否リストの完全修飾ドメイン名に比較します。名前は一致せず、Integration Server では、要求を許可すべきと判断します。これに対処するため、単純なホスト名の末尾でワイルドカード * を使用できます。または、IP アドレスを使用します。

   注:
   IP アドレスは送信偽造が困難なので、より高いセキュリティレベルが保証されます。

   次のようなパターンマッチング文字を使用すると、類似したホスト名や IP アドレスを持つ複数のクライアントを識別できます。

文字	説明	例
*	任意の数の文字に対応	r*.webmethods.com
?	任意の 1 文字に対応	workstation?.webmethods.com

7. [**ホストの追加**] をクリックします。

特定のホストからの受信要求を許可 (その他すべてを拒否)

以下の手順は、個別のポートに対する IP アクセス設定を [デフォルトで許可] に変更して、いくつかのホストを拒否する方法について説明しています。

この設定を有効にすると、このポートを介したアクセスではほとんどのホストが拒否され、少数のホストのみが許可されます。

重要:
グローバル設定を [デフォルトで拒否] に切り替える前に、グローバル設定に依存しないポートが少なくとも 1 つあり、そのポートで少なくとも 1 つのホストが許可されていることを確認してください。すべてのホストをサーバからロックアウトしてしまった場合は、適切な設定ファイルを手動で更新することで問題を解決できます。手順については、『525 ページの「誤ってすべてのホストに対して IP アクセスを拒否してしまった場合」』を参照してください。

特定のホストからの受信要求のみを許可するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの一覧**] でポートを見つけ、その [**IP アクセス**] 列にある [**許可**] または [**拒否**] リンクをクリックします。
4. [**IP アクセスモードのデフォルトを拒否に変更**] をクリックします。

   アクセスモードが変更され、許可リストにホストを追加するページが表示されます。Integration Server Administrator を使用しているマシンのホスト名と IP アドレスが既に含まれ、サーバからロックアウトされないようになっていることが確認できます。
5. [**許可リストにホストを追加**] をクリックします。
6. サーバが受信要求を受け入れるクライアントのホスト名 (`workstation5.webmethods.com` など) または IP アドレスを指定します。エントリは「`*.allowme.com, *.allowme2.com`」のようにカンマで区切ります。

   ホスト名または IP アドレスには大文字および小文字のアルファベット文字、数字 (0-9)、ハイフン (-) およびピリオド (.) を含めることができますが、スペースは含めることができません。IPv6 の場合、IP アドレスにコロン (:) およびカッコ ([]) も使用できます。

   ホストの完全修飾ドメイン名は使用しないでください。Integration Server は受信ホスト名を単純なホスト名に解決した後、単純なホスト名を許可リストの完全修飾ドメイン名に比較します。名前は一致せず、Integration Server では、要求を拒否すべきと判断します。これに対処するため、単純なホスト名の末尾でワイルドカード * を使用できます。または、IP アドレスを使用します。

   次のようなパターンマッチング文字を使用すると、類似したホスト名や IP アドレスを持つ複数のクライアントを識別できます。

文字	説明	例
*	任意の数の文字に対応	r*.webmethods.com

文字	説明	例
?	任意の 1 文字に対応	workstation?.webmethods.com

7. [**ホストの追加**] をクリックします。

特定のホストからの受信要求を拒否 (その他すべてを許可)

以下の手順は、個別のポートに対する IP アクセス設定を [デフォルトで拒否] に変更して、いくつかのホストを許可する方法について説明しています。

この設定を有効にすると、サーバはこのポートを介してほとんどのホストを許可し、少数のホストを拒否します。

特定のホストからの受信要求のみを拒否するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. [**ポートの一覧**] でポートを見つけ、その [**IP アクセス**] 列にある [**許可**] または [**拒否**] リンクをクリックします。

4. [**IP アクセスモードのデフォルトを許可に変更**] をクリックします。

5. [**拒否リストにホストを追加**] をクリックします。

6. サーバが受信要求を拒否するホストのホスト名 (`workstation5.webmethods.com` など) または IP アドレスを指定します。エントリは「`*.denyme.com, *.denyme2.com`」のようにカンマで区切ります。

   ホスト名または IP アドレスには大文字および小文字のアルファベット文字、数字 (0-9)、ハイフン (-) およびピリオド (.) を含めることができますが、スペースは含めることができません。IPv6 の場合、IP アドレスにコロン (:) およびカッコ ([]) も使用できます。

   ホストの完全修飾ドメイン名は使用しないでください。Integration Server は受信ホスト名を単純なホスト名に解決した後、単純なホスト名を拒否リストの完全修飾ドメイン名に比較します。名前は一致せず、Integration Server では、要求を許可すべきと判断します。これに対処するため、単純なホスト名の末尾でワイルドカード * を使用できます。または、IP アドレスを使用します。

   次のようなパターンマッチング文字を使用すると、類似したホスト名や IP アドレスを持つ複数のクライアントを識別できます。

文字	説明	例
*	任意の数の文字に対応	r*.webmethods.com

文字	説明	例
?	任意の 1 文字に対応	workstation?.webmethods.com

7. [**ホストの追加**] をクリックします。

誤ってすべてのホストに対して IP アクセスを拒否してしまった場合

1 つまたは複数のポートの設定を [デフォルトで拒否] に変更するときに、どのホストもサーバポートにアクセスできないようにしてしまう可能性があります。

例 1: 5 個のポートが定義済みであるとします。ポートは、グローバル IP アクセス設定を使用して設定されており、特定のホストに対する許可リストはありません。ここで、グローバル IP アクセス設定を [デフォルトで拒否] に変更すると、Integration Server はどのポートを介したどのホストのアクセスも許可しません。

例 2: 5555 と 7667 の 2 つのポートが定義済みであるとします。各ポートの IP アクセスは個別のポートで処理されています。どちらのポートにも許可リストはありません。ここで、ポート 5555 を [デフォルトで拒否] に変更し、さらに 7667 も [デフォルトで拒否] に変更すると、Integration Server はどちらのポートを介した接続もホストに許可しません。

この問題を解決するには、以下の手順に従います。

グローバル設定の IP アクセス設定のリセット

グローバル設定の IP アクセス設定をリセットするには

1. Integration Server をシャットダウンします。
2. server.cnf ファイルの編集を開始して、watt.server.hostAllow パラメータを見つけます。
3. 以下の例のように、パラメータを更新して、Integration Server にアクセスできるようにするホストの IP アドレスを指定します。

   watt.server.hostAllow=132.906.19.22

   または

   watt.server.hostAllow=2001:db8:85a3:8d3:1319:8a2e:370:7348
4. ファイルを保存して、Integration Server を再起動します。
5. [**サーバ**] > [ポート] に移動して、[**グローバル IP アクセス制限の変更**] をクリックし、必要に応じて設定を変更します。詳細については、520 ページの「すべてのポートへの IP アクセスを制御 (グローバル)」を参照してください。

個別のポートの IP アクセス設定のリセット

個別のポートの IP アクセス設定をリセットするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの一覧**] で、IP アクセス設定をリセットするポートを見つけ、ポート番号をクリックします。
4. [**<*ポートタイプ*> の詳細の表示**] ページで、そのポートに関連付けられているパッケージを確認します。たとえば、ポート 5555 は、通常、WmRoot パッケージに関連付けられています。
5. Integration Server をシャットダウンします。
6. そのパッケージの *Integration Server_directory* /instances/*instance_name*/packages/*package_name*/config ディレクトリに移動して、listeners.cnf ファイルを開きます。
7. hostAllow パラメータを見つけ、以下の例のように、このポートの使用を許可するホストの IP アドレスを示すように更新します。

```
<array name="hostAllow" type="value" depth="1">
<value>132.906.19.22</value>
</array>
```

8. ファイルを保存して、Integration Server を再起動します。
9. [**サーバ**] > [ポート] ページに移動して、各ポートの IP アクセスを必要に応じて更新します。詳細については、522 ページの「特定のホストからの受信要求を許可 (その他すべてを拒否)」および524 ページの「特定のホストからの受信要求を拒否 (その他すべてを許可)」を参照してください。

ポートから使用可能なサービスまたはWebサービス記述子の制限

ポートのアクセスモードを設定することにより、クライアントがポートを介して呼び出し可能なサービスまたはプロバイダ Web サービス記述子を制限できます。Integration Server では、次の 2 つのタイプのポートアクセスを使用できます。

- **[デフォルトで拒否]**。これは新しく作成されたポートに対するデフォルトのタイプです。このタイプは、ポートに関連付けられたリストで指定されるサービスおよびプロバイダ Web サービス記述子を除く、すべてのサービスおよびプロバイダ Web サービス記述子へのアクセスを拒否するために使用します。

[デフォルトで拒否] に設定したポートを使用すると、1 つのアプリケーションで使用するサービスセットのみがそのポートを介してアクセス可能になるようにアクセスを制限することができます。ポートを [デフォルトで拒否] に設定し、ポートに関連付けられたリストでアプリケーションのサービスを指定します。その結果、アプリケーションを使用するクライアントは、そのアプリケーション用の特定のサービスにのみアクセスできるようになります。5555 を除くすべてのポートは、最初は [デフォルトで拒否] に設定されており、限定された少数のサービスしか使用できません。

注:
`watt.server.security.createDefaultServicesOnPorts` サーバ設定パラメータを設定することで、Integration Server が内部サービスおよび組み込みサービスの事前定義済みリストを、アクセスを拒否するよう設定されたポートの許可リストに追加するかどうかを決定できます。一部の内部サービスでは実行 ACL が anonymous に設定されており、これが一部の環境でセキュリティの脆弱性となることがあります。

- **[デフォルトで許可]**。このタイプは、ポートに関連付けられたリストで明示的に拒否するサービスおよびプロバイダ Web サービス記述子を除く、すべてのサービスおよびプロバイダ Web サービス記述子へのアクセスを許可する場合に選択します。

Integration Server は、ポートを介してサービス要求を受信すると、そのポートを介してサービス要求が許可されているかどうかを確認します。サービスまたは Web サービス記述子がポートを介して呼び出し可能な場合は、サービスまたは Web サービス記述子を実行します。アクセスが拒否された場合、アクセス拒否のメッセージまたは状態をクライアントに返します。

Integration Server は、最上位サービスのポートアクセスのみを確認します。最上位サービスによって呼び出された子サービスのポートアクセスは確認しません。たとえば、serviceA が serviceB を呼び出すとします。また、ポート 5678 はデフォルトで拒否するように設定されているとします。serviceA はポートの許可リストに記載されていますが、serviceB は記載されていません。Integration Server がポート 5678 で serviceA の要求を受信すると、Integration Server はそのポートを介して serviceA を呼び出し可能かどうかを確認します。そのポートを介して serviceB が呼び出し可能かどうかは確認しません。

同様に、Integration Server は、プロバイダ Web サービス記述子のポートアクセスのみを確認します。Web サービス記述子内の操作またはハンドラサービスのポートアクセスは確認しません。

ポートアクセスコントロールには、Integration Server ネームスペースの一部ではないサービス URI を含め、任意のサービス URI を含めることができます。その結果、Web Services Stack で定義された Web サービスはポートアクセスコントロールリストに追加できます。ただし、Integration Server では、Web サービス URI のみ妥当性検査を行い、強制的に適用します。

注:
デフォルトで、Integration Server は、Axis2 に直接展開されたすべての Web サービスに対してアクセスチェックを行います。この動作は不要なことがあります。アクセスチェックを無効にするには、watt.server.portAccess.axis2 を「false」に設定します。

デフォルトで拒否するよう設定されているポートに対する次の許可リストには、Integration Server ネームスペース、フォルダ (OrderProcessing)、プロバイダ Web サービス記述子、および一部の内部サービスの一部でない 2 つの Web サービスが含まれます。

デフォルトで拒否するよう設定されているポートに対する許可リスト

許可リスト	
フォルダとサービス	削除
/ws/EchoWS.EchoWSHttpEndpoint	✕
/ws/msh/receive	✕
OrdeProcessing	✕
service:myProvideWSD	✕
wm.server.csrfguard:getCSRFSecretToken	✕
wm.server.csrfguard:isCSRFGuardEnabled	✕
wm.server.csrfguard:replaceSpecialCharacters	✕
wm.server.tx:end	✕
wm.server.tx:execute	✕
wm.server.tx:restart	✕
wm.server.tx:start	✕
wm.server:connect	✕
wm.server:disconnect	✕
wm.server:getClusterNodes	✕
wm.server:getServerNodes	✕
wm.server:noop	✕
wm.server:ping	✕

注:
デフォルトでは、Integration Server は HTTP ポートを 5555 に設定します。このポートは、ACL によって制限されていない限り、受信するすべてのサービス要求に対しアクセスを許可します。このポートは Integration Server の初期のインストールおよび設定時、また多くの開発環境においては最適ですが、実際の導入時にはこのポートをパートナやユーザが使用できるサービスへのアクセスを制限するようなポートに置き換える必要があります。

注:
ポートを介したサービスへのアクセスを制御する別の方法として、特定のクライアント認証を提示するクライアントにのみアクセスを許可するという方法もあります。詳細については、561 ページの「JAAS を使用した認証のカスタマイズ」を参照してください。

注:
Software AG Command Central では、ポートを介してアクセス可能なサービスおよび Web サービス記述子を制限するアクセスモード機能は、URL アクセスと呼ばれます。

特定のサービスへのアクセスを許可 (その他すべてを拒否)

ポートがデフォルトで拒否するように設定されている場合、Integration Server はほとんどのサービスおよびプロバイダ Web サービス記述子へのアクセスを拒否します。指定したサービスおよびプロバイダ Web サービス記述子へのアクセスは許可します。

サービス、フォルダまたはプロバイダ Web サービス記述子を 1 つずつ入力できます。さらに、Integration Server ネームスペースの一部でないサービス URI を指定できます。個々のサービスまたはフォルダを指定する代わりに、特定の実行 ACL に関連付けられたすべてのサービスおよびプロバイダ Web サービス記述子を許可できます。たとえば、カスタム Administrator ポート

を作成する場合は、Administrators ACL によって保護されているすべてのサービスまたはプロバイダ Web サービス記述子を公開することができます。

重要:
以下の手順を実行する場合は、変更する予定のポートを介してサーバにログインしないでください。この手順では、そのポートを介したすべてのサービスへのアクセスを一時的に拒否します。変更するポートを介してログインした後に、そのポートを介したすべてのサービスへのアクセスを拒否すると、サーバからロックアウトされてしまいます。したがって、別の既存のポートからログインするか、または新しくポートを作成してログインしてください。

特定のサービス、フォルダおよびプロバイダ Web サービス記述子へのアクセスを許可するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの一覧**] でポートを見つけ、その [**アクセスモード**] 列にある [**許可**] または [**拒否**] リンクをクリックします。
4. [**アクセスモードのデフォルトを拒否に設定**] をクリックします。ポートのアクセスモードが変更されます。
5. [**許可リストへのフォルダとサービスの追加**] をクリックします。
6. サービス、フォルダまたはプロバイダ Web サービス記述子の名前を入力するには、ページ左側の [**1 行につき 1 フォルダまたは 1 サービスを記述します。**] で、アクセスを許可するサービス、フォルダまたはプロバイダ Web サービス記述子の完全修飾名を入力します。項目の入力を終えるごとに Enter キーを押します。

   注:
   フォルダを指定すると、フォルダ内のすべてのサービスおよびプロバイダ Web サービス記述子へのアクセスが許可されます。

7. ACL に関連付けられているエレメントのリストから選択することにより、サービスまたはプロバイダ Web サービス記述子を指定するには、ページ右側の [**一連のフォルダおよびサービスを選択します。**] で、次の作業を実行します。

   a. [**ACL の選択**] リストで、アクセスを許可するエレメントの実行 ACL として使用する ACL を選択します。

      Integration Server Administrator によって、選択した ACL を実行 ACL として使用するすべてのエレメントが表示および選択されます。

   b. 選択したすべての項目をページ左側の許可リストに追加するには、[**選択を付加**] をクリックします。

      Integration Serverによって、選択したエントリが既存のリストに付加されます。

c. 許可リストに追加しない項目があれば、その項目の選択を解除します。複数の項目の選択を解除するには、Ctrl キーを押しながら選択を解除します。ポートに対して許可するサービスのリストに残りの項目を追加するには、[**選択を付加**] をクリックします。

   Integration Serverによって、選択したエントリが既存のリストに付加されます。

8. 上記の手順を繰り返して、このポートで使用可能なサービス、フォルダおよびプロバイダ Web サービス記述子のリストを完成させます。

9. [**追加の保存**] をクリックします。

10. [**ポートに戻る**] をクリックして、[**サーバ**] > [ポート] > [アクセスモードの編集] ページに戻ります。

特定のサービスへのアクセスを拒否 (その他すべてを許可)

ポートがデフォルトで許可するように設定されている場合、Integration Server はほとんどのサービスおよびプロバイダ Web サービス記述子へのアクセスを許可します。特定のサービスおよびプロバイダ Web サービス記述子へのアクセスは拒否します。

拒否リストには、サービス、フォルダまたはプロバイダ Web サービス記述子を 1 つずつ入力できます。さらに、Integration Server ネームスペースの一部でないサービス URI を指定できます。個々のサービスまたはフォルダを指定する代わりに、特定の実行 ACL に関連付けられているすべての項目を追加することにより、一度に複数のサービスやプロバイダ Web サービス記述子を拒否リストに追加することもできます。

特定のサービス、フォルダおよびプロバイダ Web サービス記述子へのアクセスを拒否するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. [**ポートの一覧**] でポートを見つけ、その [**アクセスモード**] 列にある [**許可**] または [**拒否**] リンクをクリックします。

4. [**アクセスモードのデフォルトを許可に設定**] をクリックします。ポートのアクセスモードが変更されます。

5. [**拒否リストへのフォルダとサービスの追加**] をクリックします。

6. サービス、フォルダまたはプロバイダ Web サービス記述子の名前を入力するには、ページ左側の [**1 行につき 1 フォルダまたは 1 サービスを記述します。**] で、アクセスを拒否するサービス、フォルダまたはプロバイダ Web サービス記述子の完全修飾名を入力します。項目の入力を終えるごとに Enter キーを押します。

   注:

フォルダを指定すると、フォルダ内のすべてのサービスおよびプロバイダ Web サービス記述子へのアクセスが拒否されます。

7. ACL に関連付けられているエレメントのリストから選択することにより、サービスまたはプロバイダ Web サービス記述子を指定するには、画面右側の [**一連のフォルダおよびサービスを選択します。**] で、次の作業を実行します。

   a. [**ACL の選択**] リストで、アクセスを拒否するエレメントの実行 ACL として使用する ACL を選択します。

      Integration Server Administrator によって、選択した ACL を実行 ACL として使用するすべてのエレメントが表示および選択されます。

   b. 選択したすべての項目をページ左側の拒否リストに追加するには、[**選択を付加**] をクリックします。

      Integration Serverによって、選択したエントリが既存のリストに付加されます。

   c. 拒否リストに追加しない項目があれば、その項目の選択を解除します。複数の項目の選択を解除するには、Ctrl キーを押しながら選択を解除します。ポートに対して拒否するサービスのリストに残りの項目を追加するには、[**選択を付加**] をクリックします。

      によって、選択したエントリが既存のリストに付加されます。

8. 上記の手順を繰り返して、このポートでアクセスを拒否するサービス、フォルダおよびプロバイダ Web サービス記述子のリストを完成させます。

9. [**追加の保存**] をクリックします。

10. [**ポートに戻る**] をクリックして、[**サーバ**] > [ポート] > [アクセスモードの編集] ページに戻ります。

デフォルトアクセスへのポートのリセット

ポートをリセットしてデフォルトアクセスに戻すと、サーバへの接続および認証に必要な標準サービスのみ使用可能になるように、ポートへの IP アクセスが変更されます。その他のサービスへのアクセスは拒否されます。

ポートをデフォルトにリセットするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. 目的のポートを [**ポートの一覧**] で見つけて、[**アクセスモード**] 列の [**編集**] をクリックします。

4. [**デフォルトのアクセス設定にリセット**] をクリックします。

   Integration Server によって、タイプが [デフォルトで拒否] に変更され、許可されるサービスのデフォルトリストが作成されます。許可されるサービスには、サーバへの接続と認証に必要な標準サービスが含まれます。

ディレクティブの使用の制御

ディレクティブは、リソースへのアクセスまたは呼び出しの方法です。Integration Server は、以下のディレクティブをサポートしています。

ディレクティブ	使用目的
admin	Administrator API のハンドラへの要求のルーティング。Administrator APIの詳細については、次を参照してください:1029ページの「Integration Server Administrator API」
default	Integration Server によって DSP ページの処理に使用されるドキュメントハンドラへの要求のルーティング
graphql	GraphQL 操作 (具体的には、GraphQL 記述子のクエリーまたは変化) の実行
invoke	サービスの実行
odata	受信 OData サービス要求の処理
rad	REST リソースが、URL テンプレートベースの方法を使用して設定され、REST API 記述子を使用して REST API として公開される必要がある場合は、REST 要求としてサブミットされたサービス呼び出しを実行します。restv2 リソースの設定方法の詳細については、『*REST Developer's Guide*』を参照してください。 **注:** restv2 リソースには rad ディレクティブを使用することをお勧めします。
rest	従来の方法を使用して REST リソースが設定されている場合は、REST 要求としてサブミットされたサービス呼び出しを実行します。REST リソースの設定方法の詳細については、『*REST Developer's Guide*』を参照してください。
restv2	URL テンプレートベースの方法を使用して REST リソースが設定されている場合は、REST 要求としてサブミットされたサービス呼び出しを実行します。restv2 リソースの設定方法の詳細については、『*REST Developer's Guide*』を参照してください。
soap	Integration Server SOAP メッセージハンドラへの要求のルーティング **注:**

ディレクティブ	使用目的
	soap ディレクティブは廃止されました。 SOAP メッセージハンドラは、バージョン 7.1.2 より前の Integration Server で提供される SOAP プロセッサにメッセージをルーティングします。このプロセッサは廃止されました。SOAP メッセージハンドラでは、Integration Server バージョン 6.5 で開発された Web サービスもサポートされています。
web	JSP ファイルへのアクセス
ws	Web サービス、具体的にはプロバイダ Web サービス記述子での操作の呼び出し

ディレクティブは次のように指定します。

```
http://host:port/directive/interface/service_name
```

次に例を示します。

```
http://localhost:5555/invoke/wm.server/ping
```

デフォルトでは、プロキシポート以外のすべての Integration Server ポートで、 上記に示したすべてのディレクティブを使用できます。プロキシポートでは、web ディレクティブ以外のすべての ディレクティブを使用できます。ただし、セキュリティ上の理由から、組織では一般的にそのビジネス要件を満たすために 必要なディレクティブのみを許可します。ファイアウォール内のユーザのみがアクセスできるポートには すべてのディレクティブを許可し、一方、 ファイアウォールの外のユーザに公開されているポートに ついてはディレクティブを制限することができます。たとえば、あるポートで内部および外部のユーザの 両方から SOAP 要求のみを 受信したい場合、そのポートでは soap ディレクティブのみを 許可します。ディレクティブの使用を特定のポートのみに制限するには、 watt.server.allowDirective パラメータを設定します (1136 ページの「watt.server.」を参照してください)。

invoke、soap および rest ディレクティブに代替名を指定することができます。たとえば、デフォルトでは、invoke ディレクティブは、URL 上で invoke (つまり、http://*host*:*port*/invoke/*folder*/*service_name*) と指定します。ユーザが invoke ディレクティブとして指定する代替語を設定できます。たとえば、ユーザが invoke ディレクティブを submit (つまり、http://*host*:*port*/submit/*folder*/*service_name*) と指定できるように設定できます。

invoke ディレクティブに代替語を指定するには、watt.server.invokeDirective パラメータを設定します (1095 ページの「サーバ設定パラメータ」を参照してください)。

soap ディレクティブに代替語を指定するには、watt.server.SOAP.directive パラメータを設定します (1095 ページの「サーバ設定パラメータ」を参照してください)。

rest ディレクティブに代替語を指定するには、watt.server.RESTDirective パラメータを設定します (1095 ページの「サーバ設定パラメータ」を参照してください)。

restv2 ディレクティブに代替語を指定するには、watt.server.RESTDirective.V2 パラメータを設定します (1095 ページの「サーバ設定パラメータ」を参照してください)。

ACL によるリソースへのアクセス制御

アクセスコントロールリスト (ACL) を使用して、Integration Server 上のパッケージ、フォルダ、ファイル、サービス、その他のエレメントへのアクセスを制御することができます。具体的には、以下に対するアクセスを制御できます。

- **クライアントが呼び出し可能なサービス**。どのグループが (どのユーザが) サービスを呼び出すことができるかを制御できます。サーバは ACL をチェックして、サービスを呼び出せるクライアントを識別するだけでなく、複数のポートレベルのチェックも実行します。これらのチェックについての説明、およびチェックを実行するためのサーバの設定については、519 ページの「リソースへのアクセスをポートごとに制御」を参照してください。
- **Integration Server Administrator、Designer、Replicator 機能などの特殊なツール**。これらの特殊な機能は、Integration Server と共に提供される Administrators、Developer および Replicators ACL で承認されています。
- **開発者がアクセスして使用できるエレメント**。各パッケージ、フォルダおよびその他のエレメントにアクセスできる開発者を詳細に制御できます。たとえば、ある開発グループは 1 つのサービスセットの作成、更新、保守ができ、別の開発グループは異なるセットにアクセスできるようにすることができます。ACL を使用すると、ある開発グループが誤って別のグループの作業を更新または破損することを防止できます。
- **サーバが提供できるファイル**。サーバは、パッケージ用の pub ディレクトリ、または pub ディレクトリのサブディレクトリにある (DSP や .htm などの) ファイルを提供することができます。.access ファイルで各ファイルに ACL を割り当てることで、これらのファイルへのアクセスを制御できます。ファイルを使用可能にする方法の詳細については、544 ページの「サーバが提供するファイルへの ACL の割り当て」を参照してください。

ここでは、ACL を使用してリソースへのアクセスを制御する方法について説明します。ポートレベルでアクセスを制御する方法については、519 ページの「リソースへのアクセスをポートごとに制御」を参照してください。

ACL について

ACL は、パッケージ、フォルダおよび (サービス、ドキュメントタイプ、仕様などの) その他のエレメントへのアクセスをグループレベルで制御します。ACL は、エレメントへのアクセスが許可されているグループ (許可グループ) と許可されていないグループ (拒否グループ) を識別します。許可グループと拒否グループの指定は、既に定義したグループから選択して行います。

アクセスには、リスト、読み取り、書き込み、実行という 4 つの種類があります。

- **リスト**ユーザはエレメントの存在を確認することができます。エレメントは Designer および Integration Server Administrator のページに表示されます。リストアクセスでは、エレメントのメタデータを表示することもできます。
- **読み取り**ユーザは、Designer および Integration Server Administrator を通じてエレメントのメインソースを表示することができます。
- **書き込み**ユーザはエレメントを編集することができます。エレメントを削除またはロックしたり、エレメントに ACL を割り当てることもできます。

- **実行**ユーザはサービスを実行することができます。さらに、DSP ファイルや .htm ファイルなどサーバが提供するファイルにアクセスすることもできます。

リスト ACL、読み取り ACL、書き込み ACL は、多くの場合開発時に開発者が使用します。また、サービスやその他のエレメントを作成、編集、保守するためにアクセスを必要とするサーバ管理者が使用することもあります。実行アクセスは、実稼動環境で広く使用されます。

ユーザがエレメントにアクセスを試みると、サーバはエレメントに関連付けられている適切な ACL (リスト、読み取り、書き込み、実行) をチェックします。

その ACL のメンバーでない限り、ACL をエレメントに割り当てることはできません。たとえば、DevTeam1 に OrderForm サービスの更新を許可する場合は、DevTeam1 ACL のメンバーである必要があります。つまり、ユーザ名が DevTeam1 ACL にリストされているグループのメンバーである必要があります。同様に、エレメントに対する ACL の割り当てを変更する場合は、既存の ACL のメンバーであり、さらにエレメントを割り当てる対象の ACL のメンバーでもある必要があります。

次の表では、それぞれのアクセスタイプがそれぞれのエレメントに対してどのような意味を持つかをまとめています。

アクセスタイプと許可される操作

要素	リスト	読み取り	書き込み	実行
パッケージ	パッケージが存在することを確認します。パッケージの内容を確認するには、エレメントへの リストアクセス権が必要です。このアクセスは、パッケージ内の他のエレメントには継承されません。	なし	なし	なし
フォルダ	フォルダが存在することを確認します。子は、特定のアクセス権が指定されていない場合にはリストアクセス権を継承します。	フォルダ自体には影響しません。子は、特定のアクセス権が指定されていない場合には読み取りアクセス権を継承します。	フォルダに対するエレメントの追加または削除を行います。また、フォルダに対する ACL の割り当てを変更します。子は、特定のアクセス権が指定されていない場合には書き込みアクセス権を継承します。	フォルダ自体には影響しません。子は、特定のアクセス権が指定されていない場合には実行アクセス権を継承します。

	アクセスタイプと許可される操作			
要素	**リスト**	**読み取り**	**書き込み**	**実行**
サービス (フロー、Java、 C、 XSLT、 アダプタサービス、Web サービス記述子など)	サービスが存在することを確認します。Designer では、サービスが非ソース情報と共に表示されます。	Designer 内のサービスのソースを確認します。	サービスの編集、ロック、アンロック、削除を行います。また、サービスに対する ACL の割り当てを変更します。	サービスを実行します。
仕様、スキーマ、フラットファイルスキーマ、ドキュメントタイプ、アダプタ通知、トリガー	エレメントが存在することを確認します。	エレメントが存在することを確認します。トリガーの場合、定義された条件を確認します。	エレメントの編集、ロック、アンロック、削除を行います。また、エレメントに対する ACL の割り当てを変更します。	なし

パッケージの複製

パッケージの複製では、パブリッシャーサーバが、ユーザに複製アクセス権が付与されていることを確認します。つまりユーザが Replicators ACL のメンバーであるかどうかを確認します。

さらに、パブリッシュを実行するユーザは、パッケージに対するリストアクセス権を持ち、Integration Server Administrator のパブリッシュページでリストを表示できる必要があります。このリスト ACL は、パッケージと共にサブスクライバサーバに送信されます。ACL は、フォルダやサービスなどパッケージ以外のネームスペースエレメントと共に送信されることはありません。

サブスクライバサーバでパッケージをインストールするには、[**受信リリースのインストール**] ページでパッケージを表示するためのリストアクセス権が必要です。つまり、ACL が必ずサブスクライバサーバ上に存在し、インストールを行うユーザは、その ACL のメンバーである必要があります。ただしこの場合、パッケージへの書き込みアクセスは必要ありません。

暗黙的および明示的な保護

エレメントが ACL によって明示的に保護されている場合、サーバは指定された ACL をチェックします。

エレメントが ACL に明示的に保護されていない場合は、次のようになります。

- ファイル以外のエレメントについては、親フォルダが ACL に保護されている場合、エレメントはフォルダの保護を継承します。フォルダが明示的に保護されていない場合、エレメントはそのフォルダの親の保護を継承します。
- ファイルについては、親フォルダが ACL に保護されている場合、ファイルはフォルダの保護を継承します。ただし、ACL によって明示的に保護されていないサブフォルダ内にファイル

がある場合、サーバは Default ACL をファイルに割り当てます。ファイルの詳細については、544 ページの「サーバが提供するファイルへの ACL の割り当て」を参照してください。

異なるパッケージ内の同一の名前のフォルダは、同じ ACL を共有します。たとえば、Finance および Marketing の両パッケージのトップレベルに MonthEnd というフォルダがある場合、フォルダ内容が異なっていても、どちらのフォルダも同一の ACL によって制御されます。

注:
トップレベルのフォルダは、親パッケージからリストアクセス権を継承することはありません。

複数のグループに属するユーザ

ユーザは 1 つまたは複数のグループのメンバーになることができます。次の表では、ユーザが単一のグループのメンバーであるときに、そのアクセス権をサーバがどう扱うかをまとめています。アクセス権がリスト、読み取り、書き込み、実行のいずれの場合でも以下のように処理されます。

ユーザが属するグループのアクセス権	パッケージ、フォルダ、その他のエレメントに対するアクセス
許可	許可
拒否	拒否
指定なし	拒否

しかし、ユーザが複数のグループのメンバーであるときはどうなるでしょうか。複数のグループのメンバーであるユーザのアクセス設定を決定するため、Integration Server は、別の強度を設定に割り当てます。具体的には、拒否は許可よりも優先し、許可は指定なしよりも優先します。

たとえば、ユーザ Smith が 3 つのグループのメンバーであり、各グループには以下に示すように FY2013 パッケージに対して異なるリストアクセスが設定されているとします。

[グループ]	FY2013 パッケージへのリストアクセス
Accounting	指定なし
Support	拒否
Corporate	許可

これらの設定の結果、Support グループの拒否設定がその他の設定よりも優先するため、Smith はパッケージに対してリストアクセスが拒否されます。

事前定義済みの ACL

サーバには次のような事前定義済みの ACL が用意されています。これらの ACL は削除できません。

- **Administrators**Administrators グループに属するユーザおよび My webMethods Administrators の役割を持つユーザにのみパッケージ、フォルダまたはその他のエレメントへのアクセスを許可し、その他のユーザはすべて拒否します。
- **Anonymous**認証されていないユーザ (ユーザ ID を指定しなかったユーザ) および My webMethods Users の役割を持つユーザのアクセスを可能にします。
- **Default**すべての認証されているユーザおよび My webMethods Users の役割を持つユーザに対して、パッケージ、フォルダまたはその他のエレメントへのアクセスを可能にします。エレメントに ACL が割り当てられていない場合、またはエレメントを含むフォルダから ACL を継承していない場合、サーバは Default ACL を使用します。エレメントに割り当てられている ACL が削除された場合、サーバは Default ACL を使用します。Default ACL は認証されているユーザのみ認可します。認証されていないユーザ (有効なユーザ ID を指定しなかったユーザ) は、Anonymous ACL によって認可されます。
- **Developers**Developers グループに属するユーザおよび My webMethods Administrators の役割を持つユーザにのみパッケージ、フォルダまたはその他のエレメントへのアクセスを許可し、その他のユーザはすべて拒否します。
- **Internal**Administrators および Developers グループに属するユーザおよび My webMethods Administrators の役割を持つユーザにのみパッケージ、フォルダまたはその他のエレメントへのアクセスを許可し、その他のユーザはすべて拒否します。サーバはこの ACL を、たとえば WmRoot や WmPublic パッケージにある、サーバの組み込みユーティリティサービスに割り当てます。この ACL をエレメントに割り当てる必要はありません。
- **Replicators**Replicator ユーザおよび My webMethods Administrators の役割を持つユーザに複製特権を許可します。

注:
wmPartnerUsers ACL など、アダプタ固有の ACL もあります。アダプタ固有の ACL の詳細については、各アダプタのマニュアルを参照してください。

ACL チェックの実行

Integration Server は次の場合に ACL のチェックを行います。

- クライアントまたは DSP が Integration Server のサービスを呼び出した場合。クライアントはブラウザのユーザ、他の Integration Server、(IS クライアント API を使用する) IS クライアントまたはカスタム HTTP クライアントのいずれかです。
- Designer を使用して、エレメントにアクセス (ACL 割り当てのリスト、作成、更新、ソースの表示、削除、変更) する場合。
- IS\x11 Administrator を使用しているときに、エレメントの ACL 割り当てをリストまたは変更する場合。

デフォルトでは、Integration Server は外部から呼び出されたサービスに対してのみ ACL のチェックを行います。外部から呼び出されたサービスとは、クライアントまたは DSP から直接呼び出されたサービスです。ただし内部から呼び出されたサービスでも、つまり Integration Server で実行中の別のサービスから呼び出された場合でも、ACL をチェックするよう設定することができます。

サービスが内部から呼び出された場合でもサーバがサービスの実行 ACL をチェックするようにするには、Designer で [**実行 ACL の適用**] プロパティを [**常に適用**] に設定します。詳細については、*webMethods Service Development Help*を参照してください。

ACL の作成

ACL を作成する場合、事前定義済みのグループからグループを選択して許可グループおよび拒否グループに使用します。

ACL を作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ACL] に移動します。
3. [**ACL の追加と削除**] をクリックします。
4. 行ごとに ACL 名を 1 つ指定します。行を分割するには Enter キーを押します。
5. [**ACL の作成**] をクリックします。

ACL へのグループアクセスの許可または拒否

新規の ACL または事前定義済みの ACL を編集することで、特定のグループには ACL へのアクセスを許可し、別のグループには ACL へのアクセスを拒否できます。内部で定義されたグループと同様、セントラルユーザディレクトリまたは LDAP を使用して外部的に定義されたグループや役割についてもアクセスを許可または拒否できます。

ACL へのグループアクセスを許可するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ACL] に移動します。[アクセスコントロールリスト] ページが表示されます。
   - [**許可**] リスト内のグループは、この ACL に関連付けられているパッケージ、フォルダ、サービス、その他のエレメントへのアクセスが明示的に許可されています。
   - [**拒否**] リスト内のグループは、この ACL に関連付けられているパッケージ、フォルダ、サービス、その他のエレメントへのアクセスが明示的に拒否されています。
3. [ACL メンバーシップ] の [**ACL を選択**] リストを使用して、グループの追加先の ACL を選択します。
4. 次のいずれかの手順に従います。

- グループまたは役割がこの ACL にアクセスすることを許可するには、[**許可**] リストで [**追加**] をクリックします。
- グループまたは役割がこの ACL にアクセスすることを拒否するには、[**拒否**] リストで [**追加**] をクリックします。

5. 表示されたダイアログボックスの [**プロバイダ**] リストを使用して、どこからユーザグループを選択するかを選択します。

   外部のユーザディレクトリが設定されていない場合、[**プロバイダ**] リストは表示されません。

6. [**役割/グループ名**] リストで、次のいずれかの操作を行います。

   - [**ローカル**] を選択した場合は、ACL へのアクセスを許可または拒否する、ローカル定義のユーザグループを選択します。
   - [**セントラル**] または [**LDAP**] を選択した場合は、[**検索**] フィールドに、役割またはグループを選択するための検索条件を入力します。[**実行**] をクリックします。ACL へのアクセスを許可または拒否する役割またはグループを選択します。

     注:
     [**検索**] フィールドの値は `watt.server.ldap.groupSearchFilter` サーバ設定パラメータの値よりも優先されます。[**検索**] フィールドに値を指定しない場合、`watt.server.ldap.groupSearchFilter` パラメータの値が使用されます。

7. [**変更内容の保存**] をクリックします。

ACL の削除

事前定義済みの ACL (Anonymous、Administrators、Default、Developers、Internal、Replicators) を除き、すべての ACL を削除することができます。パッケージ、フォルダまたはその他のエレメントに現在割り当てられている ACL も削除できます。削除された ACL に割り当てられていたエレメントに対してクライアントがアクセスを試みると、サーバはアクセスを拒否します。

パッケージ、フォルダ、サービスまたはその他のエレメントに割り当てられている ACL を削除しても、削除した ACL の名前は Integration Server によって保持されます。その結果、エレメントの情報を表示すると、関連する [**ACL**] フィールドに削除された ACL の名前が表示されます。ただしサーバはこの ACL を空の ACL として扱うのでアクセスは許可しません。

パッケージ、フォルダ、サービスまたはその他のエレメントに異なる ACL を割り当てる方法については、542 ページの「フォルダ、サービス、その他のエレメントへの ACL の割り当て」を参照してください。

異なる ACL を、サーバが提供する DSP ファイルまたは .htm ファイルに割り当てる場合は、関連する .access ファイルを更新して割り当ててください。ファイルへの ACL の割り当ての詳細については、544 ページの「サーバが提供するファイルへの ACL の割り当て」を参照してください。

ACL を削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ACL] に移動します。
3. [**ACL の追加と削除**] をクリックします。
4. ページの [**ACL の削除**] セクションで、削除する 1 つまたは複数の ACL を選択します。
5. [**ACL の削除**] をクリックします。ACL を削除するかどうか、確認を促すメッセージが表示されます。
6. [**OK**] をクリックして ACL を削除します。

デフォルト設定および継承

ここでは、新しく作成したパッケージ、フォルダ、およびその他のエレメントのデフォルト設定について、さらにフォルダの ACL 割り当てがフォルダ内のエレメントに及ぼす影響について説明します。たとえば、作成したサービスに明示的に ACL を割り当てない場合に、そのサービスの ACL 割り当てが処理される方法を説明します。通常は、以下のように処理されます。

- パッケージを作成すると、サーバはリスト ACL に Default を割り当てます(パッケージには、読み取り ACL、書き込み ACL、実行 ACL は割り当てられません)。これは、認証されているすべてのユーザがパッケージの存在を確認できることを意味します。
- 他のフォルダに含まれないトップレベルのフォルダを作成すると、サーバはリスト ACL、読み取り ACL、書き込み ACL に Default を割り当て、フォルダの実行 ACL に Internal を割り当てます。これは、認証されているすべてのユーザがパッケージの存在を確認できることを意味します。読み取り ACL および実行 ACL は、フォルダ自体にとっては意味はありません。これらは継承のみを目的として存在しています。つまり、これらの設定はフォルダ内のエレメントに継承されます。
- サブフォルダまたはその他のエレメント (サービス、スキーマ、仕様、ドキュメントタイプ、トリガーおよびその他のエレメント) を作成すると、フォルダまたはその他のエレメントに親フォルダの ACL 設定が継承されます。

この動作について、次の表でまとめています。

デフォルトで割り当てられる ACL

エレメントタイプ	リスト	読み取り	書き込み	実行
パッケージ	デフォルト	なし	なし	なし
トップレベルフォルダ	デフォルト	デフォルト	デフォルト	Internal
サブフォルダ	継承	継承	継承	継承

デフォルトで割り当てられる ACL

エレメントタイプ	リスト	読み取り	書き込み	実行
その他のエレメント	継承	継承	継承	継承

既存の ACL 割り当ての変更

特定の ACL をエレメントに割り当て、その後 ACL の割り当てを削除すると ([(継承)] に変更される)、エレメントは親フォルダの ACL を継承します。サーバには、親フォルダから継承された ACL の名前と [**(継承)**] が表示されます。トップレベルのフォルダから ACL の割り当てを削除すると、Default が適用されます。パッケージからリスト ACL の割り当てを削除すると、Default が適用されます。

重要:
Default ACL は、Everybody グループを許可グループとして、また Anonymous グループを拒否グループとして識別します。これは、エレメントに特定の ACL が割り当てられていない場合、認証されていないユーザを除くすべてのユーザが、そのエレメントにアクセスできることを意味します。リソースへの誤ったアクセスを防止するために、Default ACL に代えて、適切な ACL を割り当ててください。

フォルダの ACL 割り当てを変更すると、その中にあるエレメントの ACL 割り当てが変更されることがあります。具体的には、ACL 割り当てが [**(継承)**] となっているエレメントの割り当てが、フォルダの新しい ACL 割り当てに変更されます。既に特定の ACL が割り当てられているエレメントは変更されません。

フォルダ、サービス、その他のエレメントへの ACL の割り当て

Integration Server Administrator を使用して、ACL をフォルダ、サブフォルダまたは個別のサービスに割り当てることができます。Integration Server Administrator を使用して ACL を割り当てるときは、以下の点に留意してください。

- ACL をフォルダに割り当てると、明示的に ACL を設定していない限り、フォルダ内のすべての子はその設定を継承します。継承の詳細については、541 ページの「デフォルト設定および継承」を参照してください。
- その ACL のメンバーでない限り、ACL をエレメントに割り当てることはできません。たとえば、DevTeam1 に ProcessOrder サービスの更新を許可する場合は、DevTeam1 ACL のメンバーである必要があります。つまり、ユーザ名が、DevTeam1 ACL にリストされているグループのメンバーである必要があります。
- エレメントが、別のユーザによってロックされているか、システムロックされている場合は、そのエレメントに割り当てられている ACL を変更することはできません。ACL を割り当てることができるのは、ロックされていないエレメントか、自分がロックしたエレメントだけです。

  注:

パッケージ、仕様、ドキュメントタイプ、スキーマおよびトリガーに ACL を割り当てるには、Designer を使用します。詳細については、*webMethods Service Development Help* を参照してください。

新しい ACL や異なる ACL をフォルダまたはサービスに割り当てるには、以下の手順に従います。

ACL をフォルダまたはサービスに割り当てるには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**フォルダの表示**] をクリックします。
4. 現在のページで、ACL を割り当てるフォルダまたはサービスがリストに表示されない場合は、目的のフォルダまたはサービスを含むリストがサーバページに表示されるまで、親フォルダの名前をクリックします。
5. 適切な **ACL** のフィールドをクリックします ([**リスト ACL**]、[**読み取り ACL**]、[**書き込み ACL**]、[**実行 ACL**])。

   [**ACL 情報**] ページが表示されます。プルダウンリストを使用して、フォルダまたはサービスに割り当てる ACL を選択し、[**変更内容の保存**] をクリックします。

フォルダまたはサービスからの ACL の削除

ACL をフォルダまたはサービスから削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**フォルダの表示**] をクリックします。
4. 現在のページで、ACL を割り当てるフォルダまたはサービスがリストに表示されない場合は、目的のフォルダまたはサービスを含むリストがサーバページに表示されるまで、親フォルダの名前をクリックします。
5. 適切な **ACL** のフィールドをクリックします ([**リスト ACL**]、[**読み取り ACL**]、[**書き込み ACL**]、[**実行 ACL**])。

   [**ACL 情報**] ページが表示されます。
6. ACL 名のプルダウンメニューから [**<Default> (継承)**] を選択し、[**変更内容の保存**] をクリックします。

サーバが提供するファイルへの ACL の割り当て

サーバは、パッケージ用の pub ディレクトリまたは pub ディレクトリのサブディレクトリにあるファイルを提供することができます。Integration Server からファイルを提供する方法については、『*webMethods Service Development Help*』を参照してください。

ファイルへのアクセスを制御するには、保護するファイルを含むディレクトリに .access ファイルを配置します。.access ファイルの編集には、オペレーティングシステムの任意のツールを使用できます。

注:
.access ファイルは、DSP ファイルや HTML ファイルなど、サーバが提供するファイルへのアクセスを制御します。DSP ファイルまたは HTML ファイルで呼び出されるサービスへのアクセスを制御するには、そのサービス自体に ACL を割り当てる必要があります。詳細については、542 ページの「フォルダ、サービス、その他のエレメントへの ACL の割り当て」を参照してください。

ディレクトリにサブディレクトリがある場合は、これらのサブディレクトリには保護が継承されないので、各ディレクトリに .access ファイルを配置する必要があります。ディレクトリ内で保護する各ファイルについては、.access ファイル内に行を挿入して、ファイルおよびファイルの保護に使用する ACL を指定します。

たとえば、adminpage.dsp、home.dsp、index.htm の 3 つのファイルを含むディレクトリがあるとします。ここでは、管理者だけが adminpage.dsp ファイルにアクセスできるように、Administrators ACL を使用して adminpage.dsp ファイルを保護します。また、開発者だけが home.dsp ファイルにアクセスできるように、Developers ACL を使用して home.dsp ファイルを保護します。さらに、すべてのユーザが index.htm ファイルにアクセスできるように、index.htm ファイルに Default ACL を割り当てます。この場合は、次のレコードを .access ファイルに配置します。

```
adminpage.dsp Administrators
home.dsp Developers
index.htm Default
```

注:
上記の例では、すべてのユーザが index.htm ファイルにアクセスできるようにするので、index.htm Default は .access ファイルから削除してもかまいません。サーバは .access ファイルで識別されていないファイル、または .access ファイルを持たないディレクトリ内のすべてのファイルに対して、Default ACL を使用します。

重要:
Integration Server はパッケージをロードする際に .access ファイルをロードするので、変更内容を直ちに有効にするには、パッケージを再ロードする必要があります。

.access ファイル使用のルール

.access ファイルに入力する場合は、次のルールを守る必要があります。

- 「adminpage.dsp」の後に ACL 名を入力するというように、ファイル名だけを指定します。相対パスを指定すると、ファイルは保護されません。たとえば、pub ディレクトリ内の docs

サブディレクトリに home.dsp ファイルがあるとします (pub\docs\home.dsp)。pub ディレクトリの .access ファイルに次のエントリを追加すると、ファイルは保護されません。

```
docs\home.dsp Developers
```

代わりに次のエントリを pub\docs ディレクトリ の .access ファイルに追加します。

```
home.dsp Developers
```

- 名前入力時の大文字と小文字の区別は、ファイルシステムにおける大文字と小文字の区別によって決まります。index.dsp という名前のファイルがあるとします。Windows など大文字と小文字を区別しないシステムでは、ファイル名の文字は任意に入力できます。したがって `Index.dsp`、`INDEX.DSP` などはすべて受け入れられます。ただし UNIX など、大文字と小文字を区別するシステムでは、`index.dsp` と入力する必要があります。

ファイルからの ACL 保護の削除

ACL 保護をファイルから削除するには、以下の手順に従います。

ファイルから ACL 保護を削除するには

1. > [**シャットダウンまたは再起動**] をクリックします。
2. [**シャットダウン**] をクリックします。
3. .access ファイルを編集して、ACL 保護を削除するファイルが指定されている行を削除します。
4. サーバを再起動します。手順については、56 ページの「Integration Server の再起動 」を参照してください。

ブラックリストへのサービスの追加

Integration Server では、サービスブラックリストの使用によってサービスの呼び出しをブロックできます。ブラックリストに記載されているサービスを呼び出そうとすると、アクセス拒否エラーになります。ブラックリストに記載されているサービスは、クライアントまたはトリガーによって直接呼び出された最上位レベルのサービス、別のサービス内で呼び出されたサービス、または Java コードから Service.doInvoke() を使用して呼び出されたサービスであっても、このエラーは発生します。Designer を使用して、ブラックリストに記載されているサービスを呼び出すサービスをデバッグする場合も、ブラックリストに記載されているサービスを呼び出そうとすると、アクセス拒否エラーになります。

サービスブラックリストは、ACL を使用してサービスの表示、編集、または実行が可能なユーザを制御することとは異なります。サービスブラックリストは、WmPublic サービス、WmRoot サービスなど、Integration Server でインストールされたサービスを含め、すべてのサービスに対して使用できます。さらに、サービスブラックリストは実行のみブロックし、すべてのユーザに対してブロックします。サービスブラックリストは指定されたサービスを選択ユーザに非表示にすることはなく、特定のグループのユーザに実行を制限することもありません。パッケージの

サービスを表示した場合、ブラックリストに記載されているサービスは依然として Designer または Integration Server Administrator に表示されます。

ブラックリストに記載されているサービスを呼び出した場合、Integration Server は server.log に次のメッセージを書き込みます。

[ISS.0053.0002C] Access denied for user *username* on port *portNumber* -> *directive*/*serviceName* from *ipAddress*

注:
ユーザのクレデンシャルが正しくないか、ユーザがブラックリストに記載されているサービスを呼び出す場合を含め、ユーザがアクセスを拒否されると、Integration Server では常に上記のサーバおよびエラーログメッセージを書き込みます。

Integration Server の機能 0007 認証がデバッグレベルに設定された場合、Integration Server ではブラックリストに記載されているサービスの呼び出しを試行した後、サーバログに次のメッセージを書き込みます。

[ISS.0007.0007D] ACLManager: service execution blocked; *serviceName* has been blacklisted via "watt.server.service.blacklist property"

Integration Server では、ブラックリストの作成用にサーバ設定パラメータ watt.server.service.blacklist を提供しています。カンマ区切りリストまたはファイルを使用して、ブラックリストのサービスと、ブラックリストに記載するサービスがあるインタフェースのどちらか一方または両方を指定できます。

たとえば、サービスブラックリストに pub.client:ftp サービスを入れるには、次のように指定します。

```
watt.server.service.blacklist=pub.client:ftp
```

pub.client.ftp インタフェース (フォルダ) のすべてのサービスをサービスブラックリストに入れるには、次のように指定します。

```
watt.server.service.blacklist=pub.client.ftp
```

pub.client.ftp インタフェースをサービスブラックリストに入れるには、次のように指定します。

```
watt.server.service.blacklist=pub.client.ftp,pub.client:ftp
```

フォルダ (インタフェース) を指定すると、そのフォルダ内のサービスのみサービスブラックリストに入ります。フォルダのサブフォルダ内にあるサービスはブラックリストに入りません。たとえば、`watt.server.service.blacklist =pub.client` を指定すると、pub.client:ftp、pub.client:http、pub.client:restClient、pub.client:smtp、pub.client:soapClient、pub.client:websocket など、pub.client 内のサービスのみブラックリストに入ります。pub.client.ftp、pub.client.ldap、pub.client:oauth、または pub.client.sftp 内のサービスはサービスブラックリストに入りません。

Integration Server では、サービスブラックリストの処理時に、サービスまたはインタフェースが実際に Integration Server に存在するかどうか妥当性検査を行いません。ただし、Integration Server ではサービス名に不正な文字が含まれていないことを確認します。サービス名またはインタフェース名に不正な文字が含まれている場合、Integration Server は次のエラーをログに記録します。

[ISS.0007.0010W] ACLManager: ignored blacklist entry *entryName* as it is not a valid service or interface name.

watt.server.service.blacklist の値内、またはサービスブラックリストを含むファイルでワイルドカードは使用できません。

注:
Integration Server の機能に不可欠なサービスはブラックリストに記載しないでください。

サービスブラックリスト用ファイルの使用

サービスブラックリスト用ファイルを使用すると、組織内のサーバ間で同じブラックリストサービスのセットを簡単に指定できます。サービスブラックリスト用ファイルを使用するときには、以下の情報に留意してください。

- ファイルでは、ブラックリストの各サービスまたはインタフェースを異なる行に指定する必要があります。
- watt.server.service.blacklis 値はファイルの相対パスまたはファイルの絶対パスにする必要があります。
  - 相対パスの形式: file:{relativePath}

    相対パスの場合、Integration Server では、watt.server.homeDir パラメータに定義された Integration Server のホームディレクトリにファイルがあると想定します。watt.sever.homeDir パラメータが設定されていない場合、Integration Server ではファイルが現在の作業ディレクトリにあると想定します。このディレクトリは、Java の user.dir システムプロパティの値です。

    次に例を示します。

    ```
    watt.server.service.blacklist=file:blacklist.txt

    watt.server.service.blacklist=file:custom/blacklist.txt
    ```

  - 絶対パスの形式: file:/*directoryName*/*filename*

    次に例を示します。

    ```
    watt.server.service.blacklist=file:/opt/blacklist.cnf
    ```

- watt.server.service.blacklist の値内、またはサービスブラックリストを含むファイルでワイルドカードは使用できません。
- サービスブラックリストファイルの内容を更新した場合、変更を適用するには、Integration Server を再起動する必要があります。
- サーバ設定パラメータはカンマ区切りリスト、ファイルの相対パス、またはファイルの絶対パスに設定できます。上記の組み合わせを指定することはできません。
- Integration Server の起動時に参照先ファイルが存在しないか読み取れない場合、Integration Server はサービスをブラックリストに記載せず、起動を続行します。Integration Server は参照先ファイルに関する次の警告メッセージをログに記録します。

[ISS.0007.0009W] "ACLManager: error processing "watt.server.service.blacklist" property file.Stack trace: *stackTrace*

- リストのエントリの 1 つが構文的に無効なサービス名またはインタフェース名である場合、Integration Server ではそのエントリを無視します。

30 クライアントの認証

- 概要 550
- 基本認証 550
- ダイジェスト認証 551
- ケルベロス認証 552
- クライアント認証による認証 552
- クライアント認証とアクセスコントロール 555
- My webMethods のシングルサインオン認証 556
- SAML 認証 557

概要

Integration Server では、基本認証、ダイジェスト認証、ケルベロス認証、JWT 認証、SAML 認証、証明書ベース認証など、多くのタイプのクライアント認証を提供しています。この章では、Integration Server で利用可能なクライアントの認証のタイプを説明し、Integration Server でクライアントの認証用に設定する方法に関する情報を提供します。また、この章では、Integration Server をシングルサインオン認証用に設定し、My webMethods アプリケーションから Integration Server データにアクセスできるようにする方法についても説明します。

注:
カスタム認証スキームを使用している場合は、クライアント認証用にカスタム JAAS ログインモジュールを作成する必要があります。カスタム JAAS ログインモジュールの作成については、565 ページの「Integration Server 用のカスタム JAAS ログインモジュールの作成」を参照してください。

基本認証

基本認証を使用する場合、Integration Server はクライアントにユーザ名とパスワードを要求します。入力されたユーザ名のユーザアカウントが見つかると、Integration Server は入力されたパスワードとユーザアカウントで指定されているパスワードを比較して、ユーザ名を認証します。パスワードが正しければ、Integration Server は要求の処理を実行します。パスワードが誤っていれば、Integration Server は要求を拒否します。

クライアントがユーザ名またはパスワードを入力しない場合、Integration Server はクライアントのデフォルトのユーザアカウントを使用します。

次の表では、基本認証の使用中にクライアントが正しいユーザ名とパスワードを入力したときに Integration Server がクライアント要求をどのように処理するかをまとめています。

クライアントがユーザ名とパスワードを指定したか	ユーザ名は見つかったか	パスワードは正しいか	要求
はい	はい	はい	続行
はい	はい	いいえ	拒否
はい	いいえ	なし	拒否
いいえ	なし	なし	デフォルトのユーザアカウントを使用して続行

Integration Server は、ユーザ名とパスワードを認証キャッシュに保存します。認証キャッシュは、ユーザ名とパスワードをハッシュ形式で保存する Integration Server 内のキャッシングレイヤです。

ユーザ名とパスワードの認証に初めて成功した後に (ローカルユーザまたはセントラルユーザ/LDAP のいずれの場合でも)、Integration Server はクレデンシャルを今後の参照用に認証キャッシュに保存します。後続の認証要求時には、Integration Server は認証キャッシュ内に既存のク

レデンシャルがあるかどうかを確認します。認証キャッシュ内に既存のクレデンシャルがある場合、Integration Server は、今後クレデンシャルの妥当性検査を行いません。

注:
ユーザがパスワードを変更し、新しいパスワードでのログインに成功すると、Integration Server は旧パスワードを認証キャッシュから削除します。

認証キャッシュは、次のサーバ設定パラメータを使用して制御できます。

- **watt.server.auth.cache.enabled**認証キャッシュを有効化/無効化します。
- **watt.server.auth.cache.timeout**各キャッシュエントリが未使用状態になってから、Integration Server が認証キャッシュからエントリを削除するまでの時間をミリ秒単位で指定します。
- **watt.server.auth.cache.capacity**Integration Server によって認証キャッシュに格納される、ユーザ名とパスワードの組み合わせの数を指定します。

認証キャッシュを制御するサーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。ユーザアカウントの設定の詳細については、102 ページの「ユーザアカウントの定義」を参照してください。外部で定義されたユーザアカウントを使用することもできます。外部ディレクトリの使用、および外部ユーザアカウントを使用する場合の基本的な認証の機能については、675 ページの「セントラルユーザディレクトリまたは LDAP の設定」を参照してください。

ダイジェスト認証

Integration Server は、HTTP ヘッダーに存在するダイジェスト認証クレデンシャルを処理することにより、ダイジェスト認証をサポートしています。Web サービスヘッダーで渡されたダイジェスト認証クレデンシャルを使用して、Integration Server でホストされている Web サービスにアクセスする Web サービスコンシューマを認証します。また、Integration Server では、サードパーティの Web サービスを呼び出すときに、ダイジェスト認証クレデンシャルを送信することもできます。

パスワードダイジェストは、Nonce、作成時刻、およびパスワードのハッシュです。ダイジェスト認証を使用する場合、クライアントが Integration Server にアクセスしようとすると、Integration Server はクライアントに対し、パスワード、Nonce および作成時刻を連結してパスワードダイジェストを送信するように要求します。パスワードダイジェストを受け取った Integration Server は、データを検証してクライアントを認証します。クライアントから送信されたパスワードダイジェストが Integration Server で作成されたパスワードダイジェストと一致した場合、サーバは要求を処理します。パスワードダイジェストが一致しない場合、サーバは要求を拒否します。

ダイジェスト認証を使用するには、Integration Server Administrator でユーザを作成するときにユーザをダイジェスト認証用に設定する必要があります。また、クライアント要求の認証用にパスワードダイジェストを使用するポートを設定する必要もあります。クライアント要求の認証用にパスワードダイジェストを使用するように設定されたポートは、ユーザが認証にパスワードダイジェストを許可するように設定されている場合に限り、そのユーザからの要求を処理します。

ダイジェスト認証用にユーザを設定する方法の詳細については、104 ページの「ユーザアカウントの追加」を参照してください。

ケルベロス認証

Integration Server は、交渉認証スキームを使用してサービス要求の HTTP ヘッダーでケルベロスチケットを処理するケルベロス認証をサポートしています。Integration Server は Web サービスヘッダーで渡されたダイジェスト認証クレデンシャルを使用して、Integration Server でホストされている Web サービスにアクセスする Web サービスコンシューマを認証します。

Integration Server がケルベロスチケットを受け取ると、プリンシパル名とプリンシパルパスワードを使用して KDC にコンタクトします。ケルベロスチケットが無効ならば、Integration Server は要求を拒否します。

ケルベロスチケットが有効ならば、Integration Server はチケットに関連付けられているユーザを抽出し、ローカル Integration Server ユーザストア、セントラルユーザ、または LDAP でそのユーザを探します。Software AG は、KDC をセントラルユーザまたは LDAP のユーザディレクトリとして設定することをお勧めします。そうすると Integration Server は、クライアントが提出したケルベロスチケットに含まれるユーザを識別し認証できます。

ユーザディレクトリを設定する手順については、675 ページの「セントラルユーザディレクトリまたは LDAP の設定」を参照してください。Integration Server でケルベロスを設定する手順については、次を参照してください: 439 ページの「ケルベロスを使用するように Integration Server を設定する」

クライアント認証による認証

Integration Server のクライアント認証は、クライアントを識別する X.509 デジタル認証であり、SSL 認証に使用されます。

次に、クライアント認証による認証の設定に使用できるチェックリストを示します。

タスク	説明
1	SSL を使用するように Integration Server を設定します。詳細については、489 ページの「Integration Server における SSL の設定準備」を参照してください。
2	クライアントの署名済み認証 (信用のあるルート認証または認証チェーン) をインポートします。 信用のあるルート認証または認証チェーンは、トラストストア内に保存され、クライアントを認証するときに必要になります。ファイルを Integration Server がアクセスできるトラストストアに配置します。
3	クライアント認証を要求するか、クライアント認証を必須とするようにポートを設定します。詳細については、555 ページの「クライアント認証とポート設定」を参照してください。
4	クライアント認証をインポートしてユーザにマッピングします。 Integration Server がアクセスできる個別のファイルにクライアント認証を配置します。

タスク	説明
	詳細については、553ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。

認証マッピング

認証マッピング機能を使用すると、Integration Server 上にクライアント認証を保存し、個々の認証とユーザを関連付けることができます (たとえば、ユーザ FINANCE を識別するために認証を使用できます)。クライアントがこれらの認証のいずれかを提示した場合、Integration Server は、この認証に「マッピング」されているユーザとしてクライアントをログインさせます。

My webMethods Server では、認証とユーザを関連付けることもできます。Integration Server でセントラルユーザ管理が設定されている場合、ローカルストアでユーザを検出できなければ、Integration Server は自動的に My webMethods Server データベースをチェックします。詳細については、『*Administering My webMethods Server*』を参照してください。

重要:
クライアント認証にユーザをマッピングする場合は注意が必要です。ユーザの認証レベルが認証と適切に一致することを確認してください。

クライアント認証のインポートとユーザへのマッピング

クライアント認証は、Integration Server Administrator を使用してインポートし、ファイルに保存して特定のユーザアカウントにマッピングし、使用方法を指定します。

認証をインポートおよびマッピングする前に、以下の点を考慮してください。

- Integration Server と、クライアントとして機能するインターネットリソースとの間で SSL 接続を行う場合は、クライアントの認証もインポートする必要があります。
- Integration Server は LDAP ユーザ用の認証の読み込みをサポートしていますが、Software AG では、セントラルユーザ管理を使用して、My webMethods Server で LDAP および認証を設定することをお勧めします。

クライアント認証および CA 署名認証のインポート手順は同じで、以下のとおりです。

クライアント認証をインポートしてユーザへマッピングするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [認証] に移動します。
3. [**クライアント認証の設定**] をクリックします。
4. [**認証のパス**] フィールドに、インポートする認証が保存されているファイルのパスおよびファイル名を入力します。

注:
認証は Integration Server にアクセスできるパス上に存在する必要があります。つまり、認証は Integration Server と同じマシン上に存在する必要があります。

5. [**ユーザ**] フィールドにユーザを入力するか、 をクリックしてユーザを検索して選択します。

   [**ユーザ名**] ダイアログボックスでユーザを検索するには、次のいずれかの方法を使用します。

   - ローカルユーザを選択するため、[**プロバイダ**]リストから [**ローカル**] を選択します。認証をマッピングするローカルユーザを選択します。

     外部のユーザディレクトリが設定されていない場合、[**プロバイダ**] リストは表示されません。

   - 外部ディレクトリ (LDAP またはセントラルユーザディレクトリ) からユーザを選択するために、[**プロバイダ**] リストから、検索対象のユーザディレクトリを選択します。[**検索**] フィールドにユーザの検索条件を入力します。[**実行**] をクリックします。認証をマッピングするユーザを選択します。

6. [**用途**] リストで、この認証をインポートする目的を選択します。以下のオプションの 1 つを選択します。

   - **[SSL 認証]**Integration Server との SSL 接続を確立するときに、クライアントの認証クレデンシャルを表すために、認証を使用します。
   - **[検証]**クライアントから送信され、デジタル署名が含まれるドキュメント、メッセージまたはストリームの真偽を検証するために、認証の公開鍵を使用します。
   - **[暗号化]**Integration Server からクライアントへ送信されるドキュメント、メッセージまたはストリームを暗号化するために、認証の公開鍵を使用します。
   - **[検証および暗号化]**クライアントから送信され、デジタル署名が含まれるドキュメント、メッセージまたはストリームの真偽を検証すること、および Integration Server からクライアントに送信されるドキュメント、メッセージまたはストリームを暗号化することの両方に同じ認証を使用します。
   - **[メッセージ認証]**転送レベルではなくメッセージレベルの認証を使用する場合に (たとえば、SSL 認証情報が格納されている SOAP メッセージヘッダーを持つ Web サービスの場合)、Integration Server との SSL 接続を確立するときに、クライアントの認証クレデンシャルを表すために、認証を使用します。

7. [**認証のインポート**] をクリックします。

認証マッピングの変更

認証のマッピング先ユーザや認証の使用目的を変更することができます。

認証のマッピング先ユーザを変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [認証] に移動します。
3. [**クライアント認証の設定**] をクリックします。
4. [**現在の認証**] の [**サブジェクト CN**] 列で、マッピングを変更する認証をクリックします。
5. [**セキュリティ**] > [認証] > [クライアント認証] > [詳細] ページに移動し、[**マッピングの変更**] をクリックします。
6. [**ユーザ**] フィールドに認証をマップするユーザを入力するか、をクリックしてユーザを検索して選択します。
7. [**用途**] リストから該当する使用目的を選択します。
8. [**変更内容の保存**] をクリックします。

クライアント認証とポート設定

Integration Server では、HTTPS または FTPS 要求の場合のみクライアント認証を使用します。ポートがクライアント認証を要求するか、クライアント認証が必須であると設定された場合、Integration Server は、クライアントが Integration Server により提示されるクレデンシャルを認証すると、SSL ハンドシェーク処理中にクライアントの認証を要求します。これに続く動作は、サーバの設定や、ポートが HTTPS ポートであるか FTPS であるかによって異なります。

Integration Server では、HTTPS ポートで受信した認証についてマッピングされているユーザを自動的にチェックします。Integration Server は、FTPS ポートで受信した要求に対して、watt.net.ftpUseCertMap サーバ設定パラメータが「true」に設定されていれば、マッピングされているユーザをチェックします。

HTTPS ポートでのクライアント認証の仕組みの詳細については、180 ページの「HTTPS ポートの追加」の [**クライアント認証**] フィールドの説明を参照してください。FTPS ポートでのクライアント認証の仕組みの詳細については、199 ページの「FTPS ポートでのクライアント認証の仕組み」を参照してください。

1 つ以上のポートでクライアント認証が必須であるように設定する場合は、承認するクライアント認証をインポートして、クライアントがログインするユーザにマッピングする必要があります。

クライアント認証を必須に設定しているポートが 1 つもない場合は、クライアント認証をインポートしてユーザにマッピングし、これらの認証を提示するクライアントが自動的にユーザとしてログインできるようにすることも可能です。

クライアント認証とアクセスコントロール

クライアント認証はアクセスコントロールと共に行われます。Integration Server は、クライアントのユーザ名を識別すると、次にそのクライアントが要求するリソースへのアクセスを承認す

るかどうかを判断します。Integration Server はクライアントのグループメンバーシップを使用して、サーバリソースへのアクセスを制御します。

サーバが認証を行うクライアントの行為	要求されたリソースへのアクセスをサーバが制御する基準
サービスの呼び出し	そのクライアントが、サービスに関連付けられている実行 ACL の許可グループまたは拒否グループのいずれに属するグループのメンバーであるか
Integration Server Administrator へのアクセス	クライアントが Administrators グループのメンバーであり、したがって Administrator 特権を持っているか
Designer から Integration Server への接続	クライアントが Developers グループのメンバーであり、したがって Developer 特権を持っているか

詳細については、534 ページの「ACL によるリソースへのアクセス制御」を参照してください。

My webMethods のシングルサインオン認証

MWS (My webMethods Server) アプリケーション (webMethods Monitor、webMethods Optimize for Process など) は、Integration Server から送信されるデータへのアクセスを要求する場合があります。MWS アプリケーションがデータにアクセスできるようになる前に、MWS は Integration Server との接続を確立する必要があります。接続は次のように行われます。

- MWS から Integration Server に対してログイン要求が開始されます。
- MWS ユーザのログインクレデンシャルが Integration Server によって認証されます。
- Integration Server セッションが確立されます。

この場合、要求を開始するユーザには Integration Server クレデンシャルのセットは必要ありません。MWS ユーザデータベースに保存されているクレデンシャルを使用して、この要求を認証できます。この機能は、SSO (Single Sign-On : シングルサインオン) と呼ばれます。

重要:
SSO が動作するには、先に MWS セントラルユーザ管理が設定済みである必要があります。詳細については、*Administering My webMethods Server*を参照してください。

MWS ユーザのログインクレデンシャルの妥当性検査を行うための基本となるメカニズムには、JAAS (Java Authorization and Authentication Service) および OpenSAML 1.1 ライブラリがあります。OpenSAML ライブラリは、Integration Server に対して MWS ユーザを表す SAML アーティファクトを解決することによって、MWS ユーザを認証するために使用されます。

MWS シングルサインオンのリソース設定

このコンテキストでは JAAS および OpenSAML ライブラリの動作は透過的ですが、MWS の設定を SSO をサポートするように設定する必要があります。

- Integration Server データを要求する MWS ポートレットでは、[認証メソッド] が**ハイブリッド**に設定されている必要があります。この設定の詳細については、『*Administering My webMethods Server*』を参照してください。
- MWS シングルサインオンのリソース設定が正しい URL に設定される必要があります。

MWS シングルサインオンのリソース設定を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [リソース] に移動します。
3. [My webMethods Server のシングルサインオン] で、[MWS SAML リゾルバ URL] 設定に次の値が入力されていることを確認します。

```
https://mws-host:mws-port/services/SAML
```

SAML 認証

Integration Server では、HTTP ヘッダーにより SAML トークンを受け入れ、WS セキュリティポリシーに基づくポリシーをプロバイダ Web サービス記述子と共に使用して、SAML 認証をサポートします。Integration Server において、HTTP ヘッダーでは SAML 2.0 トークンをサポートしますが、ポリシーを使用して SAML 1.1 および SAML 2.0 のトークンをサポートします。

認証に WS-SecurityPolicy ベースのポリシーを使用し、ポリシーで SAML トークンが必要な場合は、SAML トークンを処理できるように Integration Server を設定する必要があります。Integration Server は、受信要求のプロバイダ Web サービス記述子に添付されているポリシー内の SAML トークンのみをサポートします。

プロバイダ Web サービスによって受信される受信要求メッセージでは、Integration Server は Java 認証と Integration Server の JAAS (Java Authorization and Authentication Service) ログインモジュールを使用して SAML トークンを検証できる必要があります。

注:
Integration Server は com.softwareag.is.auth.saml.issuer.SAMLIssuerUtils.generateSamlAssertion メソッドを使用して、SAML 2.0 ベアラーアサーションを生成できます。このメソッドおよび関連クラスの詳細については、*webMethods Integration Server Java API Reference*を参照してください。

SAML 認証の要件

次の表は、Integration Server が WS-SecurityPolicy に基づくポリシーで SAML トークンを処理でき、HTTP ヘッダーで SAML2 トークンを受け入れられるようにするために満たすべき要件を示しています。

要件	説明
STS (Security Token Service) プロバイダ	Integration Server が信頼する STS を決定します。クライアントは、SAML 1.1 または 2.0 トークンを生成する任意の STS プロバイダを使用できます。生成される SAML トークンは次の条件を満たす必要があります。 ■ Integration Server が HOK (Holder-of-Key) タイプの SAML アサーションを処理する場合は、クライアント認証を含む。Integration Server は、クライアント認証を使用してクライアントを特定し、Integration Server ユーザにクライアントをマッピングする。 ■ STS によって署名される。
SAML アサーションの発行者候補ごとの証明書。	各 STS のパブリックキーを含むトラストストアを作成する。トラストストアの作成の詳細については、511 ページの「トラストストアファイル」を参照してください。
信頼済み発行者の ID	信頼済み STS を Integration Server に特定する。手順については、『558 ページの「信頼済み STS を Integration Server に特定させる 」』を参照してください。
クライアント認証マッピング	Integration Server が HOK (Holder-of-Key) タイプの SAML アサーションを処理する場合は、クライアントの公開鍵が含まれており、クライアントの公開鍵を Integration Server ユーザにマッピングする必要があります。クライアント認証の設定の詳細については、552 ページの「クライアント認証による認証」を参照してください。

信頼済み STS を Integration Server に特定させる

クライアント認証用に SAML トークンを含めるには、Integration Server が SAML トークンを処理できるよう、設定する必要があります。要件の 1 つは、Integration Server が信頼する STS (Security Token Service) プロバイダを特定することです。

信頼済み STS を Integration Server に特定させるには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [SAML] に移動します。
3. [**SAML トークン発行元の追加**] をクリックします。
4. 次のフィールドに情報を指定ます。

パラメータ	指定内容
[発行元名]	Integration Server が SAML アサーションを受け入れて処理する SAML トークン発行者の名前。この値は、SAML アサーションの [**Issuer**] フィールドの値と一致する必要があります。 Integration Server はこのページで設定されていない発行者からの SAML アサーションを拒否し、サーバログに次のようなメッセージを記録します。 `2010-06-09 23:35:38 EDT [ISS.0012.0025E] [セキュリティ] > [SAML] 画面で 発行元が設定されていないため、発行元 "SAMPLE_STS" からの SAML アサーションを拒否しています。`
[トラストストアエイリアス]	トラストストアのテキスト ID。SAML トークン発行者の公開鍵を含みます。Integration Server は、既存のトラストストアエイリアスで [**トラストストアエイリアス**] のリストを生成します。
[認証エイリアス]	トラストストアエイリアスに関連付けられる認証のテキスト ID。Integration Server は、選択されたトラストストアの認証エイリアスで [**認証エイリアス**] のリストを生成します。
[クロックスキュー]	Integration Server をホストするマシンと SAML トークン発行元との間のクロックの違い (ミリ秒単位)。負以外の数を指定します。 たとえば、Integration Server マシンのクロックと発行元のクロックとの間で 3 秒の時間差がある場合、3001 ミリ秒のスキューを指定できます。一定のバッファを許容するため、3200 や 4000 など、多少高めのスキューを指定できます。 SAML アサーションの解析後、Integration Server はタイムスタンプをミリ秒に変換し、そのミリ秒を使用してすべての妥当性検査を実行します。これにより、Integration Server で実行される SAML の妥当性検査は、発行元と Integration Server での異なるタイムゾーンの使用をサポートします。 NotBefore クレームの妥当性検査を実行する場合、Integration Server はアサーションにある NotBefore 時刻 (この時点ではミリ秒で表現されます) からクロックスキューを差し引きます。次に、Integration Server は調整済みの NotBefore 時刻を Integration Server をホストするマシンの現在の時刻と比較し、Integration Server マシンの時刻が調整済みの NotBefore 時刻より小さいことを確認します。 NotAfter クレームの妥当性検査を実行する場合、Integration Server は現在の時刻 (ミリ秒で表現されたもの) からクロックスキューを差し引きます。次に、Integration Server は調整済みの現在の時刻とアサーションにある NotAfter 時刻を比較し、NotAfter 時刻が調整済みの現在の時刻と等しいかそれより大きいことを確認します。

5. [**変更内容の保存**] をクリックします。

転送レベルで SAML2 トークンを受け入れる

Integration Server では、HTTP ヘッダーに SAML2.0 トークンを入れることができるため、あらゆるタイプのサービスで SAML トークンを使用できます。そのため、他のセキュリティプロバイダとの統合が可能です。

この機能を使用するには、SAML アサーションの発行元を信用のある SAML 発行元のリストに追加します。詳細については、558 ページの「信頼済み STS を Integration Server に特定させる」を参照してください。Integration Server:

要求を送信するクライアントは、wmIS-SAML2-Assertion というカスタム HTTP ヘッダーを含め、Base64 でエンコードされた SAML2.0 アサーションをヘッダー値として送信する必要があります。

Integration Server が、カスタムヘッダー wmIS-SAML2-Assertion を持つ HTTP 要求を受信して、ヘッダー内に Base64 でエンコードされた SAML2.0 アサーションを検出すると、Integration Server は Base64 からデコードし、アサーションの妥当性検査を行います。アサーションの妥当性検査が成功したら、Integration Server は SAML2.0 アサーションから NameID に一致する Integration Server ユーザを検索します。Integration Server は最初に Integration Server に定義されたローカルユーザをチェックし、次にセントラルユーザまたは LDAP を検索します。Integration Server が NameID に一致するユーザ名を検出すると、Integration Server はセッションにそのユーザ名を使用します。それ以外の場合は、Integration Server は Integration Server is_jaas.cnf の IS_Transport ログインコンテキストにおける SamlAssertLoginModule モジュールの defaultUserName オプションに定義されたユーザを使用します。defaultUserName オプションが「Default」に設定されている場合、Integration Server はデフォルトユーザアカウントを使用します。デフォルトユーザアカウントは Anonymous ACL を持つリソースにアクセスできます。

31 JAAS を使用した認証のカスタマイズ

- 概要 562
- Integration Server での JAAS の使用 562
- JAAS 設定ファイル 562
- プラグ接続可能な認証モジュール (PAM) 564
- Integration Server 用のカスタム JAAS ログインモジュールの作成 565
- JAAS カスタムログインモジュールの例 566

概要

この章では、カスタム JAAS ログインモジュールを設定して、Integration Server 認証で使用する方法について説明します。

Integration Server での JAAS の使用

JAAS では、カスタムログインモジュールを展開するための標準ベースのメカニズムを提供します。JAAS を使用すると、独自のカスタムログインモジュールを作成して、Integration Server 認証の処理に使用することができます。

Java コードベースのセキュリティを拡張する JAAS フレームワークを使用することで、Integration Server 認証をカスタマイズし、認証プロセスで複数のログインモジュールを呼び出すことができます。JAAS を使用すると、次のことを指定できます。

- カスタムログインモジュールの呼び出し順序
- ログインモジュールが必須であるか、またはオプションであるか
- ログインモジュールから制御元アプリケーションに制御を戻すタイミング

JAAS を使用してカスタムログインモジュールを実装する場合は、次の作業を行う必要があります。

- ログインモジュールを作成する
- JAAS 設定ファイルの適切なログインコンテキスト内でログインモジュールを設定する
- ログインモジュールを含む JAR ファイルを Integration Server のクラスパスに追加する

> **注:**
> JAAS カスタムログインモジュールが処理するのは、Integration Server ユーザの認証のみです。JAAS 機能を Integration Server 認可に使用することはできません。Integration Server は ACL を使用して認可を行います。詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。

JAAS 設定ファイル

JAAS 設定ファイルは、JVM 内で使用するログインモジュールの選択を制御するファイルです。Integration Server は、JAAS 設定ファイルとして *Integration Server_directory* \instances*instance_name*\config\is_jaas.cnf を使用するように JVM を設定します。

JAAS ログインモジュールのセットは、「ログインコンテキスト」にグループ化されます。各ログインコンテキストでは、ログインモジュールの完全な名前、オプションのパラメータおよび成功か失敗かに基づいて実行するアクションを指定します。これらの指定は、REQUIRED、REQUISITE、SUFFICIENT および OPTIONAL に分類されます。ログインに成功するためには、一連のログインコンテキストがすべて成功する必要があります。

JAAS 設定ファイルには、次の内容がリストされます。

- 使用可能なログインコンテキスト

- 実行されるログインモジュール
- モジュールの実行順序
- モジュールの失敗時に実行するアクションを決定する設定

次に、Integration Server のデフォルト JAAS 設定ファイルの一部を示します。IS_Transport および WSS_Message_IS というログインコンテキストがあることがわかります。Integration Server の JAAS カスタムログインモジュールには、次の認証があります。

- トランスポートレベル認証。IS_Transport ログインコンテキストで指定します (次のコードの濃い灰色部分)。
- Web サービス用のメッセージレベル認証。WSS_Message_IS ログインコンテキストで指定します。Integration Server のメッセージレベル認証については、『*Web Services Developer's Guide*』に記述されています。

注:
JAAS 設定ファイルには、この他にもログインコンテキストが含まれています。ここでは、IS_Transport および WSS_Message_IS についてのみ説明します (次の is_jaas.cnf のコードセグメントを参照)。

```
IS_Transport { /*
          com.wm.app.b2b.server.auth.jaas.X509ValidatorModule requisite; */
          com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
          com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;
          com.wm.app.b2b.server.auth.jaas.SamlOSGiLoginModule requisite; /* * The
          DefaultLoginModule contains logic that provide special * default handling
for
          Software AG products so please leave * this module as the last module of
this
          login context.*/ com.wm.app.b2b.server.auth.jaas.DefaultLoginModule
          requisite;};
```

```
WSS_Message_IS { /* * Please do
          not rearrange the following SoftwareAG * login modules; add your login
modules
          before or after * these three modules */
          com.wm.app.b2b.server.auth.jaas.SamlAssertLoginModule requisite;
          com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
          com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;};
```

プリインストールされたログインモジュール

IS_Transport ログインコンテキストには、X509LoginModule、BasicLoginModule、SamlOSGiLoginModule など、デフォルトのログインモジュールが含まれています。

各ログインモジュールは、それぞれが必要とするクレデンシャルが存在するかどうかを、最初にチェックします。モジュールにクレデンシャルがある場合は、そのクレデンシャルを使用します。モジュールにクレデンシャルがない場合は、「false」の値を返します。

たとえば、BasicLoginModule はユーザ名およびパスワードの取得を試みます。どちらも見つからない場合はこのモジュールから「false」の値が返され、モジュールの実行が失敗します。このモジュールは、「requisite」を指定して IS_Transport 内にリストされています。このため、

ログイン全体が成功するには、ログインコンテキスト内のモジュールのいずれかが成功する必要があります。

X509ValidatorModule

X509ValidatorModule はオプションの機能であるため、上記の IS_Transport ログインコンテキストで、このモジュールはコメント化されています。このモジュールを有効にすると、指定した X509Certificate チェーンに対してパスの妥当性検査が実行されます。また、次に示すように check_crl_status パラメータおよび crl_url パラメータを追加すると、CRL (Certificate Revocation List : 認証取り消しリスト) チェックを実行するように検査を設定できます。

```
IS_Transport
{
com.wm.app.b2b.server.auth.jaas.X509ValidatorModule requisite
check_crl_status=true
crl_url="file:///C:\\webMethods\\sec\\crl\\lh.crl";
com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;
com.wm.app.b2b.server.auth.jaas.SamlOSGiLoginModule requisite;
};
```

crl_url には、上記のようにファイル URL を指定するか、 または有効な CRL URL (http://myca.com/crl/lh.crl など) を指定できます。

認証パスの妥当性検査の場合、このモジュールは、 要求を受信したポートに関連付けられているトラストストアの使用を 試みます。ポートのトラストストアが特定されなかった場合、このモジュールは、 このモジュールは Integration Server のデフォルト送信トラストストアを使用します。次に示すように、 モジュールの「trustStore_alias」プロパティを使用すると、この設定を上書きして 独自の Integration Server トラストストアエイリアスを指定できます。

```
com.wm.app.b2b.server.auth.jaas.X509ValidatorModule requisite
trustStore_alias="my trustStore";
```

プラグ接続可能な認証モジュール (PAM)

JAAS カスタムログインモジュールを実装すると、旧バージョンの Integration Server で使用された認証カスタマイズの PAM (Pluggable Authentication Module : プラグ接続可能な認証モジュール) アプローチよりも優先されます。PAM による認証のカスタマイズは、Integration Server バージョン 8.0 で廃止されています。

PAM は、次の場合に JAAS ログインモジュール BasicLoginModule および X509LoginModule の拡張として引き続き Integration Server バージョン 8.0 以降で動作します。

- 「基本」認証メカニズムに PAM を登録した場合は、BasicLoginModule の拡張として扱われる
- 「X509」認証メカニズムに PAM を登録した場合は、X509LoginModule の拡張として扱われる

PAM が登録されているときに、対応する JAAS ログインモジュール (BasicLoginModule または X509LoginModule) が呼び出された場合、まずそのモジュールに指定されたクレデンシャルで認

証が試行されます。この認証が成功しなかった場合は、次に PAM が呼び出され、もう一度同じユーザの認証が試行されます。

重要:
JAAS 設定ファイルで BasicLoginModule または X509LoginModule を指定するコード行を削除すると、対応する認証メカニズムに登録済みの PAM がすべて無効になります。

注:
PAM による Integration Server クライアント認証のカスタマイズに関する情報については、『*webMethods Integration Server Administrator's Guide* バージョン 8.0』を参照してください。

Integration Server 用のカスタム JAAS ログインモジュールの作成

ここでは、Integration Server 用のカスタム JAAS ログインモジュールを作成および展開する方法の概要を説明します。

SagAbstractLoginModule の拡張

- `com.softwareag.security.jaas.login.SagAbstractLoginModule` を拡張する
- 次の抽象メソッドを実装する

```
initConfiguration()
authenticate(com.softwareag.security.jaas.login.SagCredentials
sagcredentials)
```

Integration Server はクレデンシャルを収集し、 com.softwareag.security.jaas.login.SagCredentials オブジェクトに クレデンシャルを移入します。 Integration Server は次に、適切なログインコンテキスト (ISTransport など) 上で JAAS ログインを開始します。JAAS フレームワークは、ログイン コンテキストに関連付けられた ログインモジュールの設定を検索します。各ログインモジュールが順に呼び出されて、初期化およびログインが行われます。 この処理は SagAbstractLoginModule によって行われ、このモジュールは ログインモジュール固有の初期化およびログインを派生クラスに委任します。たとえば、モジュールで JAAS ログインが呼び出されると、 SagAbstractLoginModule は com.softwareag.security.jaas.login.SagCredentials を 渡してログインモジュールの `authenticate(sagcredentials)` を呼び出します。

ログインモジュールの `authenticate` メソッドは `SagCredentials` オブジェクトからクレデンシャルを抽出し、認証を実行する必要があります (詳細については、`SagCredentials` および `SagAbstractLoginModule` の Javadoc を参照してください)。

認証が成功すると、`SagCredentials` に正しいユーザ名が移入されます。`SagCredentials` に含まれる現在のユーザ名が使用するユーザ名と違う場合は、`sagcredentials.setUserName(string)` メソッドを使用して、ユーザ名を更新します。

commit() の実装

ログインコンテキストの設定によっては、異なるユーザ名で複数のモジュールが成功する可能性があります。Integration Server には、異なるユーザ名で複数のログインモジュールが成功する

場合に実行するアクションを決定する、デフォルトメカニズムが存在しません。このような状況を回避するため、Integration Server のログインモジュールには次の commit メソッドが実装されています。

```
public boolean commit() throws LoginException {
createUserPrincipal = "true";
super.commit();
return true;
}
```

ここで、 createUserPrincipal は SagAbstractLoginModule の メンバー変数です。メソッド super.commit() は、 SagAbstractLoginModule の commit() メソッドを 示しています。この commit() メソッドにより、 SagCredentials からユーザ名が抽出され、 Subject 内に SagUserPrincipal オブジェクトが存在しない場合にのみ、 SagUserPrincipal が作成されます。

ログインモジュールには、上記の commit() メソッドを実装する必要があります。

IS_Transport に含まれる複数のログインモジュールが成功した場合、プリンシパルを作成するのは commit() を呼び出した最初のモジュールのみです。したがって、commit() を実装すれば、ニーズを満たすように JAAS 設定ファイル内のログインモジュールの順序を並べ替えることができます。Subject 内に複数のプリンシパルがある場合、Integration Server はインデックス 0 のプリンシパルを使用します。

JAAS がユーザを認証できる場合、JAAS は javax.security.auth.Subject を返します。Integration Server は、この JAAS サブジェクトを現在のセッションに追加します。次の呼び出しを行うと、このサブジェクトを抽出できます。

```
com.wm.app.b2b.server.InvokeState.getCurrentSession().getCurrentJaasSubject()
```

Integration Server のクラスパスへの JAR ファイルの配置

ログインモジュールを含む JAR ファイルをコンパイルしたら、Integration Server のクラスパスに配置します。パッケージの code\jars\static フォルダのいずれかに JAR ファイルを配置すると、Integration Server は静的クラスパスにファイルをロードします。Integration Server が起動したら、Integration Server Administrator の [製品情報] ページを使用して、ログインモジュールを含む JAR ファイルが Integration Server のクラスパスに表示されることをチェックします。

JAAS 設定ファイルの変更

JAAS 設定ファイル (*Integration Server_directory* \instances*instance_name*\config\is_jaas.cnf) を開き、ログインモジュールを指定する IS_Transport コンテキストにコード行を追加します。

JAAS カスタムログインモジュールの例

ここでは、サンプル JAAS カスタムログインモジュールの構成について説明し、コードの重要部分にはコメントを付けて解説します。説明のため、ログインモジュールは意図的に単純化されています。特定のユーザを認証するようにハードコードされています。

Integration Server 用の JAAS ログインモジュール: サンプルコード

次に、簡単な Integration Server JAAS ログインモジュールの Java コードを示します。番号が振られているコード部分については、モジュールの後の表で概要を説明します。

```
package samples.login;
```

1.

```
import javax.security.auth.login.LoginException;
import com.softwareag.security.jaas.login.SagAbstractLoginModule;
import com.softwareag.security.jaas.login.SagCredentials;
import com.softwareag.security.jaas.principals.SagUserPrincipal;
import com.wm.app.b2b.server.UserManager;
```

2.

```
public class TestLoginModule extends SagAbstractLoginModule {
```

3.

```
private String userId;
```

4.

```
public boolean abort() throws LoginException {
         userId = null;
         return true;
      }
```

5.

```
public boolean commit() throws LoginException {
            if(userId != null)
      {
                  createUserPrincipal = "true";
                  super.commit();
                  return true;
            } else {
                  return false;
            }
      }
```

6.

```
protected void initConfiguration(){
            this.userId = null;
      }
```

7.

```
public boolean authenticate(SagCredentials userCreds) throws
LoginException
      {
            String username = userCreds.getUserName();
            if(username == null || username.length()==0) {
```

```
                return false;
            }
            if(userCreds.getPassword() == null) {
                return false;
            }
            String password = new String(userCreds.getPassword());
            if(password == null || password.length()==0) {
                return false;
            }
            if(username.equals("bob") && password.equals("123") &&
                UserManager.getUser(username) != null)
            {
                userId = username;
                return true;
            }  else {
                return false;
            }
      }
      public boolean logout() throws LoginException {
            userId = null;
            return true;
      }
}
```

JAAS カスタムログインモジュール: コードの説明

次の表では、上記のサンプルモジュールに含まれるカスタム JAAS ログインモジュールコードの重要部分をまとめています。

コード番号	説明
1.	インポートするクラス。最初が Oracle の Java クラス、次の 3 つが *Software AG_directory* `\common\lib` にある sin-common.jar の Software AG クラスです。
2.	JAAS カスタムログインモジュールを `SagAbstractLoginModule` の拡張として指定します。
3.	認証から取得するユーザ ID。
4.	ログインコンテキストの認証全体が失敗した場合、このメソッドによってログインが中断されます。 ログインモジュールの認証試行が成功すると、このメソッドは、もともと保存されていた状態情報をすべてクリーンアップします。
5.	このコードブロックは、565 ページの「commit() の実装」で説明した commit の実装を示します。commit メソッドは、このログインモジュールによって SagUserPrincipal が作成された場合は「true」を返し、ログインモジュールを無視する必要がある場合は「false」を返します。
6.	カスタム JAAS ログインモジュールを初期化します。

コード番号	説明
7.	指定した `com.softwareag.security.jaas.log.SagCredentials` オブジェクトからクレデンシャルを抽出して、ユーザの認証を試みます。コミットフェーズの間、`SagAbstractLoginModule` は、`SagCredentials` オブジェクトで指定されたユーザ名を使用して、プリンシパルを作成します。`SagCredentials` オブジェクトに含まれる現在のユーザ名が使用するユーザ名と違う場合は、`sagcredentials.setUserName(string)` メソッドを使用して、ユーザ名を正しいユーザ名で更新します。 単純化のため、メソッドではハードコードしたユーザ名およびパスワードが指定されています。

JAAS 設定ファイル: サンプルモジュール

次に、ここで説明したサンプルログインモジュールを指定するために、JAAS 設定ファイル (*Integration Server_directory* \instances*instance_name*\config\is_jaas.cnf) の IS_Transport コンテキストに追加するコード行を示します (`requisite` を指定)。

```
IS_Transport
{
 samples.login.TestModule requisite;
 com.wm.app.b2b.server.auth.jaas.X509LoginModule requisite;
 com.wm.app.b2b.server.auth.jaas.BasicLoginModule requisite;
 com.wm.app.b2b.server.auth.jaas.SamlOSGiLoginModule requisite;
...
};
```

32 マスターパスワードと送信パスワード

- 概要 572
- 送信パスワードの管理 573
- 送信パスワードおよびマスターパスワードファイルのバックアップ 573
- マスターパスワードの変更 574
- マスターパスワードの有効期間の変更 575
- configPassman.cnf ファイルについて 575
- 送信パスワード設定の使用 576
- マスターパスワード設定の使用 577
- マスターパスワードを紛失または忘れた場合の対処方法 580
- 起動時にマスターパスワードまたは送信パスワードに問題が生じた場合 580
- 電子メールリスナーとパッケージの複製 584

概要

通常動作の一部として、Integration Server はリモート Integration Server、プロキシサーバ、データベースなどのアプリケーションおよびサブシステムに接続します。Integration Server がクライアントとして機能している場合、これらのシステムに接続するために、システムごとに「送信パスワード」と呼ばれるパスワードの入力が要求されます。Integration Server は、送信パスワードを使用して、Integration Server 自体を識別するか、他のシステムに対して認証します。

Integration Server をアプリケーションまたはサブシステム (例: データベース) に接続するように設定する場合は、Integration Server がデータベースサーバ接続のために送信する必要があるパスワードを指定します。その後 Integration Server ユーザがデータベースを必要とする要求を作成すると、Integration Server が設定済みのパスワードをデータベースサーバに送信し、データベースサーバに接続します。

注:
キーストアまたはトラストストアエイリアスを作成すると、常に、Integration Server によって自動的に、キーストアまたはトラストストアのパスワードが送信パスワードとして保存されます。詳細については、509 ページの「Integration Server におけるキーストアとトラストストアの使用」を参照してください。

これらの送信パスワードは、保護するために Integration Server によって暗号化されます。デフォルトで、送信パスワードは Password-Based Encryption (PBE) テクノロジ (PKCS5 とも呼ばれる) を使用して暗号化されます。この暗号化方式では、暗号化鍵または指定したマスターパスワードを使用する必要があります。暗号化された送信パスワードはファイル内に保存されます。

注:
フローサービスでは pub.security.outboundPasswords サービスを使用して、セキュアリソースにアクセスするために送信パスワードを保存および取得することもあります。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

マスターパスワードも暗号化され、デフォルトでファイル内に保存されます。ただし、パスワードをファイル内に保存すると、何者かによってファイルがアクセスされパスワードが復号化される可能性が生じます。セキュリティを強化するために、代わりに Integration Server の設定でマスターパスワードの入力をサーバ起動時に促すようにすることができます。

注:
マスターパスワードファイル (使用時) および送信パスワードファイルを保護するには、ファイルにオペレーティングシステムの Administrator 権限を割り当てます。

上記のように、送信パスワードは他のエンティティに対して認証するために Integration Server によって使用されます。これに対し、受信パスワードは Integration Server に対して認証するためにユーザおよび他のサーバによって使用されます。受信パスワードは一方向ハッシュとして保存されます。受信パスワードの設定については、101 ページの「ユーザとグループの管理」を参照してください。

ここでは、送信パスワードの管理方法について説明します。

送信パスワードの管理

Integration Server を初めてインストールすると、Integration Server は PBE を使用して送信パスワードを暗号化するように設定され、90 日の有効期間で「manage」というマスターパスワードが設定されます。

マスターパスワードおよび有効期間を変更するには、Integration Server Administrator の [**セキュリティ**] > [送信パスワード] ページを使用します。また、Integration Server Administrator を使用して、マスターパスワードまたは送信パスワードの紛失や破損などの不測の事態においてもマスターパスワードとすべての保存済み送信パスワードをリセットすることができます。

その他の設定を変更するには、configPassman.cnf ファイルを編集する必要があります。設定内容を次に示します。

- 送信パスワードの暗号化方式。
- Integration Server で使用するマスターパスワード取得方式。Integration Server は、マスターパスワードをファイルに保存することも、サーバ起動時に入力を促すこともできます。

次の表は、実行できるタスクと説明の参照先を示しています。

変更対象	**以下を参照してください。**
マスターパスワード	574 ページの「マスターパスワードの変更」
マスターパスワードの有効期間	575 ページの「マスターパスワードの有効期間の変更」
送信パスワードに使用する暗号化方式	576 ページの「送信パスワード設定の使用」
送信パスワードストアの場所	576 ページの「送信パスワード設定の使用」
マスターパスワードの繰り返し制限。つまり、以前に使用したパスワードを再使用できるようにするまでの期間。	577 ページの「マスターパスワード設定の使用」
マスターパスワードストアの場所	577 ページの「マスターパスワード設定の使用」
すべての送信パスワードとマスターパスワード	582 ページの「マスターパスワードと送信パスワードのリセット」

送信パスワードおよびマスターパスワードファイルのバックアップ

他の重要なシステムファイルに対して行うように、送信パスワードおよびマスターパスワードの保管のために Integration Server が使用するファイルを定期的にバックアップする必要があります。これらのファイルは Integration Server インスタンスのホームディレクトリ (*Integration Server_directory* /instances/*instance_name*) にあります。

以下の表は、送信パスワードおよびマスターパスワードを保管するファイルはを示しています。

ファイル	コンテンツ
config/txnPassStore.dat	暗号化された送信パスワードを保存
config/empw.dat	暗号化されたマスターパスワードを保存
config/configPassman.cnf	送信パスワードの設定を指定
config/passman.cnf	configPassman.cnf の編集不可バージョン

以上のファイルは、同時にバックアップおよび復元する必要があります。送信パスワードストアまたはマスターパスワードストアの名前または場所を変更する場合は、バックアップ手順によってファイルが適切にバックアップされることを確認してください。

マスターパスワードの変更

Integration Server を初めてインストールしたときのマスターパスワードは「manage」です。セキュリティ上の理由により、インストールの直後にマスターパスワードを変更してその後も定期的に変更する必要があります。また、人事異動がある場合にも変更が必要です

マスターパスワードのデフォルトの有効期間は 90 日です。期限日が近づくと、Integration Server によって Integration Server 上にパスワード有効期限の状態が表示され、マスターパスワードを変更しなくてはならない時期であることを示す警告メッセージがサーバコンソールに送信されます。電子メール通知を行うように Integration Server が設定されている場合は、この情報を伝える電子メールメッセージも Integration Server によって設定アドレスに送信されます。

マスターパスワードを変更する場合は、以下の点に留意してください。

- 送信パスワードをマスターパスワードと常に同期させるために、Integration Server は、マスターパスワードが変更されている間は、送信パスワードの保存および取得の要求を処理しません。そのため、エイリアスが多数ある場合は、パフォーマンスの低下を避けるためにピーク時を外してマスターパスワードの変更を行うことを検討してください。
- マスターパスワードを紛失した場合は、581 ページの「パスワードを復元できるかどうかの判断」を参照してください。

マスターパスワードを変更するには

1. Integration Server Administrator を開きます。
2. [**セキュリティ**] > [送信パスワード] に移動します。
3. [**マスターパスワードの更新**] をクリックします。
4. 現在のパスワードを入力してから、新しいパスワードを入力し確認します。
5. [**パスワードの変更**] をクリックします。

マスターパスワードの有効期間の変更

マスターパスワードのデフォルトの有効期間は 90 日です。現在の有効期限日を確認するには、[**セキュリティ**] > [送信パスワード] ページを表示します。

有効期間は推奨されるパスワード変更間隔です。 有効期限日までにマスターパスワードを変更しなかった場合、Integration Server は既存のパスワードによる動作を続行します。

お勧めできませんが、期間に 0 を指定することができます。この設定により、パスワードの期限がなくなり、Integration Server Administrator またはサーバログに警告が送信されなくなります。

マスターパスワードの有効期間を変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [送信パスワード] に移動します。
3. [**有効期間の更新**] をクリックします。
4. 新しい有効期間の日数を入力し、[**更新**] をクリックします。最大期間は 366 日です。

configPassman.cnf ファイルについて

configPassman.cnf ファイルには、Integration Server Administrator に含まれていない送信パスワード暗号化に対する追加設定が含まれています。ファイルは、いくつかのプロパティで構成され、デフォルトで一部がコメント化されています。

重要:
configPassman.cnf ファイルには、関連ファイル passman.cnf があります。configPassman.cnf ファイルに変更を加えると、Integration Server の初期化時に Integration Server によって自動的に変更内容を反映するように passman.cnf が更新されます。passman.cnf を直接更新しないようにしてください。

出荷時の configPassman.cnf ファイルでは、送信パスワードはファイル *Integration Server_directory* /instances/*instance_name*/config/txnPassStore.dat に保存され、PBE で暗号化されるように指定されています。さらに、マスターパスワードが *Integration Server_directory* /instances/*instance_name*/config/empw.dat ファイルに保存されることも指定されています。他の設定を指定するためのプロパティはコメント化されています。

オプションの設定を変更する場合は configPassman.cnf ファイルを編集する必要があります。ファイルでは必ず以下の項目を指定します。

- 送信パスワードの暗号化方式。
- 送信パスワードを含むファイルの場所
- Integration Server で使用するマスターパスワード取得方式

ここでは、configPassman.cnf ファイルの詳細と送信パスワードおよびマスターパスワード設定の変更方法について説明します。

送信パスワード設定の使用

configPassman.cnf ファイルには、送信パスワードファイルの名前、場所および暗号化を指定するプロパティが含まれています。

重要:
Integration Server を初めて起動して設定する前に、送信パスワードとマスターパスワードの制御方法を決定しておきます。Integration Server を設定した後でこれらの設定を変更すると、マスターパスワードと送信パスワードの同期が外れることがあります。

configPassman.cnf ファイルを使用してマスターパスワードの設定を変更する方法については、577 ページの「マスターパスワード設定の使用」を参照してください。

configPassman.cnf ファイルをよく知らない場合は、次に進む前に575 ページの「configPassman.cnf ファイルについて」を読んでください。

送信パスワードファイルの名前および場所の制御

送信パスワードファイルのデフォルトのファイル名および場所は、サーバインスタンスのホームディレクトリにある config/txnPassStore.dat ファイルの outbound.password.store.field.fileName プロパティによって決定されます。

outbound.password.store.field.fileName プロパティの値を変更するときは、以下の情報に留意してください。

- outbound.password.store.field.fileName プロパティは、configPassman.cnf ファイル内に常に存在し、コメント解除されている必要があります。
- 絶対パスまたは相対パスを指定できます。
- パス名には、スラッシュ (/) のみを使用します。バックスラッシュ (\) はサポートしません。

送信パスワードファイルのファイル名または場所を変更するには

1. テキストエディタを使用して、次のファイルを開きます。

   Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. `outbound.password.store.field.fileName` プロパティを見つけて編集します。

3. configPassman.cnf ファイルを保存して閉じます。

送信パスワードファイルの暗号化の制御

configPassman.cnf ファイルには、送信パスワードファイルの使用可能な各暗号化メソッド用のプロパティが含まれています。送信パスワードファイルのデフォルト暗号化メソッドは 256 ビットキーを使用する AES アルゴリズムです。

送信パスワードファイルが使用する暗号化メソッド用のプロパティのコメントを解除し、使用しない暗号化メソッド用のプロパティをコメント化します。プロパティのいずれか 1 つのみを必ずコメント解除しておきます。

送信パスワードファイルの暗号化メソッドを変更するには

1. テキストエディタを使用して、次のファイルを開きます。

   Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. 次の表のプロパティの 1 つをコメント解除して、送信パスワードファイルが使用する暗号化を指定します。

プロパティ	説明
`default.encryptor=EntrustAes`	256 ビットキーを使用する AES アルゴリズム。これがデフォルトです。
`default.encryptor=EntrustPbePlus`	PBE。
`default.encryptor=Base64`	セキュアではない Base64 エンコーディング。
`default.encryptor=None`	セキュアではないクリアテキスト。

3. 残りのプロパティのコメント化し、プロパティのいずれか 1 つのみをコメント解除しておきます。

4. configPassman.cnf ファイルを保存して閉じます。

マスターパスワード設定の使用

configPassman.cnf ファイルには、マスターパスワードの設定を変更できるプロパティが含まれています。デフォルトでは、マスターパスワードはサーバインスタンスのホームディレクトリの config/empw.dat ファイルに保管されています。必要に応じて、サーバ初期化時にマスターパスワードの入力を促すように Integration Server を設定することができます。

ファイル内のマスターパスワードの保管の詳細については、578 ページの「ファイルへのマスターパスワードの保存」を参照してください。

詳細については、578 ページの「サーバ初期化時のマスターパスワード入力」を参照してください。

configPassman.cnfファイルをよく知らない場合は、次に進む前に575ページの「configPassman.cnfファイルについて」を読んでください。

重要:
Integration Server を初めて起動して設定する前に、送信パスワードとマスターパスワードの制御方法を決定しておきます。Integration Serverを設定した後でこれらの設定を変更すると、マスターパスワードと送信パスワードの同期が外れることがあります。

configPassman.cnfファイルを使用して送信パスワードの設定を変更する方法については、576ページの「送信パスワード設定の使用」を参照してください。

ファイルへのマスターパスワードの保存

configPassman.cnf は、マスターパスワードストアの設定に使用できるプロパティを提供します。

ファイルにマスターパスワードを設定するには

1. テキストエディタを使用して、次のファイルを開きます。

   Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. 次の表に示すプロパティを設定します。

プロパティ	説明
`master.password.storeInFile`	Integration Server がファイルにマスターパスワードを保存する (true) かどうかを制御します。「true」に設定します。
`master.password.field.fileName`	マスターパスワードストアの場所 デフォルトは config/empw.dat です。 キーストアを指定するときは、スラッシュ (/) のみ使用します。バックスラッシュ (\) はサポートされません。
`master.password.field.repeatLimi`	同じパスワードを再利用する前に必要とされるパスワード変更の回数です。デフォルトは 3 です。

3. configPassman.cnf ファイルを保存して閉じます。

サーバ初期化時のマスターパスワード入力

configpassman.cnf ファイルは、サーバ初期化時にマスターパスワード用プロンプトを使用するためのプロパティを提供します。

以下のプロパティは、Integration Server でサーバ初期化時にパスワードの入力を促す場合、つまり master.password.storeInFile に「`false`」を指定した場合のみ使用します。

サーバ初期化時にマスターパスワードの入力を促すように Integration Server を設定するかどうかを決定する際、以下の情報に留意してください。

- サーバ初期化時にマスターパスワードの入力を促すには、`master.password.storeInFile` プロパティは「`false`」に設定されている必要があります。
- Integration Server は、custom_wrapper.conf の `wrapper.java.additional.205=-Djava.awt.headless=true` プロパティによって制御されたヘッドレスモードで実行される必要があります。

  サーバ起動時に Integration Server でマスターパスワードの入力を促す場合、custom_wrapper.conf がに `wrapper.java.additional.205=-Djava.awt.headless=false` を含めるように編集する必要があります。

  custom_wrapper.conf ファイルの編集の詳細については、47 ページの「設定ファイル custom_wrapper.conf の設定の変更」を参照してください。

  注:
  ヘッドレスモードで Microservices Runtime を実行するには、 *Integration Server_directory* /bin/setenv.bat(sh) の JAVA_CUSTOM_OPTS プロパティを編集して、プロパティ java.awt.headless=true を含めます。Java のシステムプロパティを Microservices Runtime に渡す方法については、『 *Developing Microservices with webMethods Microservices Runtime*』を参照してください。
- 以下の場合は、サーバ初期化時に Integration Server でマスターパスワードの入力を促すように設定することができません。
  - Integration Server を Windows サービスとして実行する。
  - Integration Server を UNIX 上のバックグラウンドアプリケーションとして実行する。
- サーバ初期化時に Integration Server でパスワードの入力を促さない場合は、configpassman.cnf ファイルの `master.password.field.useGUI` および `master.password.field.attemptsLimit` プロパティをコメント化します。

サーバ初期化時に Integration Server でマスターパスワードの入力を促すように設定するには

1. テキストエディタを使用して、次のファイルを開きます。

   Integration Server_directory /instances/*instance_name*/config/configPassman.cnf

2. `master.password.field.useGUI` および `master.password.field.attemptsLimit` がコメント化されていないことを確認します。これらのパラメータがファイルに存在しない場合は、パラメータを追加してください。

3. 次の表に示すプロパティを設定します。

プロパティ	説明
`master.password.storeInFile`	Integration Server がファイルにマスターパスワードを保存するかどうかを制御します。「false」に設定します。
`master.password.field.usgeGUI`	Integration Server のどこでマスターパスワードの入力を促すかを指定します。「true」に設定して、パスワードの入力を促すメッセージがポップアップウィンドウに表示されるようにします。このメソッドを選択すると、Windows の [スタート] メニューから Integration Server を開始できます。 「false」に設定して、パスワードの入力を促すメッセージがサーバコンソールに表示されるようにします。このメソッドを選択すると、Windows の [スタート] メニューからは Integration Server を開始できません。 デフォルトは「true」です。
`master.password.field.attemptsLimit`	Integration Server が要求を拒否するようになる前に許可されるログイン試行の失敗の回数。デフォルトは 3 です。

4. configPassman.cnf ファイルを保存して閉じます。

マスターパスワードを紛失または忘れた場合の対処方法

PBE で送信パスワードを暗号化するように Integration Server が設定されている場合は、Integration Server でマスターパスワードが保持され、そのパスワードが送信パスワードを暗号化するための鍵になります。新しい暗号化鍵に変更するときは必ずマスターパスワードを入力します。さらに、一部のインストールでは、Integration Server 初期化時にマスターパスワードの入力を促し、パスワードが入力されないと Integration Server が後述の「セーフモード」で開始するように Integration Server が設定されています。

したがって、マスターパスワードを紛失または忘れた場合は、状況に応じてマスターパスワードを復元するかリセットする必要があります。詳細については、581 ページの「パスワードを復元できるかどうかの判断」を参照してください。

起動時にマスターパスワードまたは送信パスワードに問題が生じた場合

起動時にマスターパスワードや送信パスワードに問題があることが Integration Server で検知されると、セーフモードになります。これは、問題の診断と修正を行うことができる特別なモードです。

Integration Server がセーフモードになると Integration Server Administrator が表示されますが、Integration Server は外部リソースには接続されません。

マスターパスワードまたは送信パスワードに問題があるためにセーフモードになっている場合は、Integration Server Administrator の [サーバの統計情報] ページの左上隅に次のメッセージが表示されます。

```
サーバがセーフモードで実行中です。マスターパスワードの健全性チェックが失敗しました。無効な
マスターパスワードが提供されました。
```

重要:
セーフモードになっている場合、**すべての送信パスワードのリセット**を行うタスクの一部として送信パスワードがリセットされている場合を除き、送信パスワードを設定または変更しないでください。

これらのパスワードに問題がある場合、パスワードを復元またはリセットすることによって問題を修正できます。修正の方法は、パスワードの問題によって異なります。Integration Server が自動的にセーフモードに移行する理由はいくつかあります。

パスワードが破損したか同期が外れている
マスターパスワードファイル、送信パスワードファイルのいずれかまたは両方のファイルが破損している可能性があります。これらのファイルの同期が外れている可能性もあります。送信パスワードファイルの内容を暗号化する鍵がマスターパスワードファイルの鍵でない場合、ファイルの同期が外れています。いずれのケースも、詳細については、581 ページの「パスワードを復元できるかどうかの判断」を参照してください。

マスターパスワードの入力に誤りがある
サーバ起動時にマスターパスワードの入力を促されたときに誤ったパスワードを入力したため、セーフモードに移行しています。このケースが考えられる場合、Integration Server をシャットダウンして再起動し、マスターパスワードの入力を促されたときに正確に入力します。

プラットフォームロケールが変わった
OS ロケールまたはデフォルトのエンコーディングが変更されると、送信パスワードファイルおよびマスターパスワードファイルが Integration Server から読み取れなくなる場合があります。この理由から、Software AG では、Integration Server をインストールして開始した後にはプラットフォームロケールを変更しないことをお勧めします。

パスワードを復元できるかどうかの判断

以下のいずれかに該当する場合、パスワードを復元できます。

- マスターパスワードおよび送信パスワードがファイルに保存されており、両方のファイルと passman.cnf ファイルの最新バックアップがある。
- Integration Server がマスターパスワードの入力を促すように設定されており、送信パスワードファイルと passman.cnf ファイルの最新バックアップがあり、そのバックアップ用のマスターパスワードがわかる。

パスワードをリセットしなければならないのは、以下のいずれかに該当する場合です。

- マスターパスワードと送信パスワードがファイルに保存されているが、マスターパスワードファイル、送信パスワードファイルおよび passman.cnf ファイルの最新バックアップがない。

- Integration Server がマスターパスワードの入力を促すように設定されているが、送信パスワードファイルと passman.cnf ファイルの最新バックアップがない。
- Integration Server がマスターパスワードの入力を促すように設定されているが、マスターパスワードを紛失したか忘れた。

マスターパスワードファイルと送信パスワードファイルの復元

これらのファイルを復元する前に、必ず 581 ページの「パスワードを復元できるかどうかの判断」を読んで、復元できるかリセットが必要かを判断してください。

マスターパスワードファイルと送信パスワードファイルを復元するには

1. 復元が必要なファイルを判断します。

   マスターパスワードがファイルに保存されていない、つまり Integration Server によってサーバ起動時にマスターパスワードの入力が促される場合は、送信パスワードファイルと passman.cnf ファイルのみを復元できます。それ以外の場合は、マスターパスワードファイル、送信パスワードファイルおよび passman.cnf ファイルをバックアップから復元する必要があります。

2. ファイルの名前と場所を判断します。

   passman.cnf ファイルは、常にサーバインスタンスのホームディレクトリ (*Integration Server_directory* \instances*instance_name*) にある config/passman.cnf です。デフォルトでは、マスターパスワードファイルは config/empw.dat であり、送信パスワードファイルは config/txnPassStore.dat です。これらのファイルのシステム上の場所が確かでない場合は、ファイル config/configPassman.cnf で調べてください。このファイルの使用の詳細については、575 ページの「configPassman.cnf ファイルについて」を参照してください。

3. Integration Server をシャットダウンします。

4. 更新ファイルを該当ディレクトリにコピーします。

5. Integration Server を再起動します。

注:
マスターパスワードファイル (使用する場合)、送信パスワードファイルおよび passman.cnf ファイルは、必ず同時にバックアップおよび復元してください。

マスターパスワードと送信パスワードのリセット

これらのパスワードをリセットする前に、必ず581 ページの「パスワードを復元できるかどうかの判断」を読んでパスワードのリセットが本当に必要なのか、または、パスワードを回復できるかどうかを判断してください。

リセット手順により、保存された送信パスワードがクリア (消去) され、マスターパスワードが「manage」にリセットされます。さらに、すべてのアプリケーションおよびサブシステムのパ

スワードを Integration Server Administrator のそれぞれの設定ページで手動で再入力する必要があります。

保存された送信パスワードとマスターパスワードをリセットするには

1. Integration Server が実行中でないときは、それを起動します。サーバ初期化時にマスターパスワードの入力を促すように Integration Server を設定している場合は、なんらかの値を入力します。

   Integration Server がセーフモードに移行します。このモードは Integration Server Administrator ですが、外部リソースには接続されていません。

2. [**セキュリティ**] > [送信パスワード] に移動します。

3. [**マスターパスワードの更新**] をクリックします。

4. [**すべての送信パスワードのリセット**] をクリックします。

   パスワードをリセットするかどうかを確認する警告ページが Integration Server に表示されます。

5. [**パスワードのリセット**] をクリックします。

   パスワードを本当にリセットするかどうかを確認するメッセージが Integration Server に再度表示されます。

6. [**OK**] をクリックします。

   この手順で保存された送信パスワードがクリアされ、マスターパスワードが「manage」に変更されます。

7. [**送信パスワード**] ページで [**パスワードの変更**] をクリックし、マスターパスワードを「manage」以外に変更します。

8. Integration Server を再起動します。

   サーバにパスワードが保存されなくなったアプリケーションおよびサブシステムへの接続を Integration Server が試行するときにエラーメッセージが表示されます。

9. 各アプリケーションまたはサブシステムの設定ページに移動し、Integration Server がそのアプリケーションまたはサブシステムに接続するために必要なパスワードを再入力します。確認すべきページは、リモートサーバエイリアス、クラスタ設定、JDBC 接続プール、電子メールリスナー、LDAP サーバ、プロキシサーバ、Broker 設定、WmDB を定義する画面などです。

電子メールリスナーとパッケージの複製

リスナーに関連付けられているパッケージをエクスポートする場合、リスナーに関する情報がパッケージと共に送信されます。ただし、電子メールリスナーの場合、すべてのリスナー設定情報が送信先 Integration Server に送信されるわけではありません。具体的には、電子メールリスナーが電子メールサーバに接続するために使用する送信パスワードは送信されません。そのため、送信先 Integration Server 上のリスナーが電子メールサーバに接続しようとすると、接続が失敗します。リスナーがポートのリストに表示されますが、リスナーは有効になりません。また、サーバコンソールにエラーメッセージも表示されます。

ポートを有効にするには、Integration Server Administrator の [**サーバ**] > [ポート] > [電子メールクライアント設定の編集] ページに移動し、[**パスワード**] フィールドを更新して電子メールサーバへの接続に必要なパスワードを指定します。

電子メールリスナーに関連付けられたパッケージを 6.5 の Integration Server から 6.5 より前の Integration Server にエクスポートする場合は、電子メールリスナーがまったく複製されません。パッケージをインストールしたら、6.5 より前の Integration Server でリスナーを手動で再設定する必要があります。

33 CSRF ガードによる Integration Server のセキュリティ確保

- CSRF とは 586
- Integration Server で CSRF 攻撃を防止する方法 586
- CSRF ガードの用語について 586
- Integration Server での CSRF ガードの設定 588
- Integration Server で CSRF ガードを設定する場合の制限事項 590

CSRF とは

クロスサイトリクエストフォージェリ (CSRF) は、Web サイトおよび Web アプリケーションに対する最も一般的な攻撃の 1 つです。CSRF 攻撃は、悪意のある要求を含む Web ページをユーザが誤ってロードしたときに発生します。この Web ページから、そのユーザの ID と特権を使用して、悪意のある要求が Web サイトまたは Web アプリケーションに送信され、設定の変更やサービスの呼び出しなどの望ましくないアクションが実行されます。

Web アプリケーションで、認証されているユーザからの入力に基づいてアクションが実行されるが、個別のアクションを認可することがユーザに要求されない場合、そのアプリケーションは CSRF 攻撃に対して脆弱になります。つまり、Web ブラウザに格納された Cookie によって、Web アプリケーションに対する認証を行う場合、悪意のある HTTP または HTTPS 要求を気付かずにアプリケーションに送信する可能性があります。

Integration Server で CSRF 攻撃を防止する方法

Integration Server では、CSRF ガード機能を使用して CSRF 攻撃を防止します。Integration Server では、Integration Server Administrator または他のクライアントアプリケーションから承認要求を受信したときに、セッションごとに 1 つの CSRF セキュアトークンを作成することによって、CSRF 攻撃を防止します。この CSRF セキュアトークンは、セッションが期限切れになるまで Integration Server によって後続の要求に追加されます。セッションが終了すると CSRF トークンは期限切れになります。

要求を送信すると、Integration Server によって要求内のトークンの存在と有効性が検証され、要求内のトークンはセッション内のトークンと比較されます。要求内にトークンがない場合、または要求内のトークンがセッション内のトークンと一致しない場合、要求はIntegration Server によって終了されます。また、このイベントは Integration Server によって CSRF 攻撃の可能性としてサーバログおよびセキュリティ監査ログに記録されます。

Integration Server で CSRF ガードを有効または無効にするには、Integration Server Administrator を使用します。

Integration Server では、以下について CSRF セキュアトークンが挿入および検証されます。

- ダイナミックサーバページ (DSP) の Web ブラウザからの HTTP 要求
- invoke、rest、または restv2 ディレクティブの HTTP 要求
- Ajax XMLHttpRequest

CSRF ガードの用語について

Integration Server で CSRF ガードを設定する前に、Integration Server での CSRF ガードに関して使用される以下の用語を理解しておくと、役立ちます。

- **除外するユーザエージェント** ユーザエージェント値は、HTTP 要求の `User-Agent` HTTP ヘッダーに対応するストリングです。除外するユーザエージェントは、Integration Server によって CSRF ガードが適用されないユーザエージェントです。つまり、これらの除外するユーザエージェントからの要求では、CSRF トークンは Integration Server によってチェックされません。

ユーザエージェントを正規表現として指定できます。正規表現に使用できるワイルドカード文字は、アスタリスク (*) のみです。エントリを区切るには、1 行につき 1 つのユーザエージェントを入力します。行を分割するには Enter キーを押します。

次に例を示します。

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

```
Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us)
AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341
Safari/528.16
```

```
*Mozilla*
```

- **ランディングページ**Integration Server パッケージのホームページは、ランディングページと呼ばれます。Integration Server によってランディングページ内の CSRF セキュアトークンはチェックされませんが、そのページのトークンが挿入されます。これらのランディングページからの CSRF セキュアトークンを含むその後のすべての要求は、Integration Server によって保護されます。

  ランディングページは正規表現として指定できません。エントリを区切るには、1 行につき 1 つのランディングページを入力します。行を分割するには Enter キーを押します。

  次に例を示します。

  ```
  MyPackage/index.dsp
  ```

  ```
  MyPackage/index.html
  ```

- **非保護 URL**Integration Server で CSRF セキュアトークンをチェックする必要のない URL は、非保護 URL と呼ばれます。Integration Server では、このフィールドに指定されていないすべての URL から送信される要求には、CSRF セキュアトークンが含まれている必要があります。

  非保護 URL として DSP ページを指定した場合、Integration Server によってそのファイルの CSRF セキュアトークンは挿入されません。保護されているページにこの DSP ページからアクセスしようとすると、拒否アクションの設定に応じて、Integration Server によってエラーが発行されるか、Integration Server Administrator のホームページにリダイレクトされます。

  非保護 URL を正規表現として指定できます。正規表現に使用できるワイルドカード文字は、アスタリスク (*) のみです。エントリを区切るには、1 行につき 1 つの URL を入力します。行を分割するには Enter キーを押します。

  次の表は、[**非保護 URL**] テキスト領域 URL の例です。

例	Integration Server による CSRF セキュアトークンのチェックの対象外
`MyPackage/abc.dsp`	MyPackage パッケージ内の abc.dsp ページ
`MyPackage/*`	MyPackage パッケージ内のすべてのページ
`invoke/pub.math:addInts`	pub.math:addInts サービスを呼び出す要求

例	Integration Server による CSRF セキュアトークンのチェックの対象外
`invoke/pub*`	pub で始まるすべてのサービスを呼び出す要求
`invoke/*`	すべての呼び出し要求

- **拒否アクション**要求に CSRF セキュアトークンが含まれていないか、無効な CSRF セキュアトークンが含まれていることが検出されたときに Integration Server で実行するアクション。次のように動作するように Integration Server を設定できます。
  - ユーザを Integration Server Administrator のホームページまたは Integration Server によって CSRF 攻撃が検出されたという警告を表示する Web ページにリダイレクトします。
  - エラーを発行して要求を終了します。

Integration Server での CSRF ガードの設定

Integration Server で CSRF ガードを設定するときは、以下の点に留意してください。

- Integration Server で CSRF ガードを有効または無効にする場合は、Web ブラウザをリフレッシュして変更内容を有効にする必要があります。
- 以下の状況では、Integration Server によって CSRF 攻撃からの保護は提供されません。
  - Anonymous ACL が割り当てられたサービスを呼び出すための、実行アクセスを持つユーザからの要求。
  - Integration Server セッションがタイムアウトした後の、Integration Server Administrator またはクライアントアプリケーションからの要求。このような場合、Integration Server によってユーザがリダイレクトされるか、エラーが発行されます。続行するには、Web ブラウザをリフレッシュする必要があります。
  - セッションの作成時に使用されたユーザエージェントとは異なるユーザエージェントからの要求。

Integration Server で CSRF ガードを設定するには

1. Integration Server Administrator を開きます。
2. [**セキュリティ**] > [CSRF ガード] > [CSRF ガード設定の編集] に移動します。
3. [**有効**] チェックボックスをオンにして、Integration Server で CSRF ガードを有効にします。
4. [**除外するユーザエージェント**] テキスト領域で、CSRF ガードを適用しないユーザエージェントを入力します。

   ユーザエージェント値は、HTTP 要求の `User-Agent` HTTP ヘッダーに対応するストリングです。

ユーザエージェントを正規表現として指定できます。正規表現に使用できるワイルドカード文字は、アスタリスク (*) のみです。エントリを区切るには、1 行につき 1 つのユーザエージェントを入力します。行を分割するには Enter キーを押します。

5. [**ランディングページ**] テキスト領域で、Integration Server 内のパッケージのランディングページのリストを入力します。Integration Server によってランディングページ内の CSRF セキュアトークンはチェックされませんが、そのページのトークンが挿入されます。これらのランディングページからの CSRF セキュアトークンを含むその後のすべての要求は、Integration Server によって保護されます。

   ランディングページは正規表現として指定できません。エントリを区切るには、1 行につき 1 つのランディングページを入力します。行を分割するには Enter キーを押します。

6. [**非保護 URL**] テキスト領域で、Integration Server で CSRF セキュアトークンをチェックする必要のない URL を入力します。

   非保護 URL を正規表現として指定できます。正規表現に使用できるワイルドカード文字は、アスタリスク (*) のみです。エントリを区切るには、1 行につき 1 つの URL を入力します。行を分割するには Enter キーを押します。

   注:
   [**非保護 URL**] テキスト領域でランディングページを指定しないでください。ランディングページの URL を [**ランディングページ**] と [**非保護 URL**] の両方のテキスト領域で指定すると、ランディングページオプションが優先され、Integration Server によってそれらのページで CSRF セキュアトークンはチェックされません。

7. [**拒否アクション**] のオプションから、要求に CSRF セキュアトークンが含まれていないか、無効な CSRF セキュアトークンが含まれていることが検出されたときに Integration Server で実行するアクションを選択します。

選択項目	目的
[**エラー**]	アクセス拒否エラーを発行して要求を終了します。これがデフォルトです。 [**エラー**] を選択すると、要求に CSRF セキュアトークンが含まれていないか、無効な CSRF セキュアトークンが含まれていることが検出されたときに、Integration Server によって以下のエラーが発行されます。 `Access Denied.Invalid CSRF secure token.` このエラーメッセージは、 Integration Server によって CSRF 攻撃が検出されたことを示します。 このエラーメッセージは、Integration Server によって以下の状況でも発行されます。 ■ Integration Server セッションが期限切れになった。 ■ CSRF ガードを有効にした後に Web ブラウザをリフレッシュしていない。

選択項目	目的
	■ 同じ Integration Server に接続している別のユーザが Integration Server を再起動した。 このような場合、続行するには Web ブラウザをリフレッシュします。
[リダイレクト]	以下のようにユーザをリダイレクトします。 ■ ユーザが Integration Server Administrator で DSP ページにアクセスしたときに CSRF の脅威が検出された場合、Integration Server Administrator のホームページにユーザをリダイレクトします。 ■ invoke、rest、または restv2 ディレクティブを含む URL またはクライアント要求で CSRF の脅威が検出された場合、Integration Server によって CSRF 攻撃が検出されたという警告を表示する Web ページにユーザをリダイレクトします。サービスを実行するには、ユーザは [**続行**] をクリックする必要があります。 **注:** クライアントアプリケーションが text/html をコンテンツタイプとして受け入れる場合のみ、Integration Server はユーザをこのページにリダイレクトします。クライアントアプリケーションが text/html を受け入れない場合、Integration Server はアクセス拒否エラーを返します。

8. [**変更内容の保存**] をクリックし、Web ブラウザをリフレッシュして変更内容を有効にします。

Integration Server で CSRF ガードを設定する場合の制限事項

- Integration Server で CSRF ガードを有効または無効にする場合は、Web ブラウザをリフレッシュする必要があります。
- 各サーバ上の ISInternal 機能エイリアスが同じデータベースを指し示す Integration Server の非クラスタグループの一部として Integration Server が実行される場合、CSRF ガード機能は使用できません。
- document.location や window.location.href などの JavaScript Location オブジェクトを使用するカスタム DSP ページには、Integration Server によって CSRF セキュアトークンは挿入されません。これらのページは手動で更新する必要があります。

  JavaScript 変数 *_csrfTokenNm_*、*_csrfTokenVal_*、*is_csrf_guard_enabled* および *needToInsertToken* を定義する必要はありません。ただし、これらの変数を使用するには、*Integration Server_directory* \instances*instance_name*\packages\WmRoot\csrf-guard.js を DSP にインポートする必要があります。
- これらのリンクが DSP を指し示す場合に限り、Integration Server によって DSP のリンクに CSRF セキュアトークンが挿入されます。これらのリンクが DSP を指し示していない場合、CSRF セキュアトークンが含まれるようにリンクを手動で更新する必要があります。

たとえば、次のコードが DSP に含まれていたとします。

```
<a href="/invoke/wm.sap.Transaction/viewAs?type=xml</a>
```

これを次のコードに置き換える必要があります。

```
<a href="/invoke/wm.sap.Transaction/viewAs?type=xml&secureCSRFToken=%value
secureCSRFToken%"></a>
```

DSP での CSRF ガードの使用の詳細については、『*Dynamic Server Pages and Output Templates Developer's Guide*』を参照してください。

34 webMethods Enterprise Gatewayの設定

- 概要 595
- Enterprise Gatewayの動作 595
- Enterprise Gateway Serverと内部サーバ間のバージョンの相互運用性 598
- 従来のサードパーティ製プロキシサーバに対するEnterprise Gatewayの利点 600
- サービス拒否保護について 600
- モバイルアプリケーション保護について 601
- モバイルデータの同期について 601
- SQLインジェクション対策について 602
- アンチウイルススキャンフィルタについて 603
- カスタムフィルタについて 604
- Enterprise Gateway設定内のクラスタリング 605
- Enterprise Gatewayの設定 605
- Enterprise Gatewayポートの設定 607
- 内部サーバのEnterprise Gateway Serverへの接続 612
- Enterprise Gateway登録ポートへの接続の表示 616
- Enterprise Gateway Serverでのクライアント認証の実行 617
- Enterprise Gatewayルールの使用 618
- 警告オプションの指定 625
- サービス拒否攻撃の防止 627

■ モバイルアプリケーションの使用の制御 630

■ Enterprise Gateway についてよく寄せられる質問 632

概要

Integration Server が内部ファイアウォールの内側にあり、DMZ を介した外部クライアントとの通信が許可されていない場合は、webMethods Enterprise Gateway を設定することで、モバイルアプリケーションなどの外部クライアントからの要求をこのサーバで処理できます。

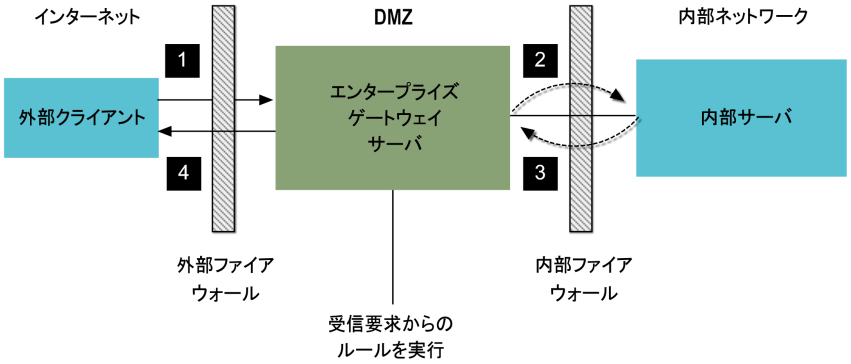
Enterprise Gateway 設定では、内部ファイアウォールの内側に存在する Integration Server は内部サーバと呼ばれます。通常、Enterprise Gateway Server と呼ばれる別の Integration Server を DMZ に配置します。Enterprise Gateway Server は外部クライアントと内部サーバの媒介として機能し、内部サーバとそのアプリケーション、サービスおよびデータを悪意のある攻撃から保護します。Enterprise Gateway Server は、保証付きデリバーを含めて、通常の Integration Server が処理するほとんどすべての要求をサポートします。

デフォルトでは、すべてのユーザ妥当性検査およびトランザクション処理は内部サーバで実行されますが、ユーザ妥当性検査を実行するように Enterprise Gateway Server を設定することができます。Enterprise Gateway Server 上でルールを設定して、内部サーバに渡される前に要求をフィルタできます。

重要:
Enterprise Gateway を使用するには、webMethods Enterprise Gateway ライセンスが必要です。Enterprise Gateway のライセンスでは、Integration Server の一部の機能のみを使用できます。独自のホスト Integration Server に Enterprise Gateway をインストールしてください。Enterprise Gateway をホストする Integration Server に他の製品をインストールしないでください。

Enterprise Gateway の動作

次の図は、外部クライアント要求が Enterprise Gateway 設定で処理される仕組みを示したものです。



外部クライアントは要求を Enterprise Gateway Server に送信します (1)。Enterprise Gateway Server は各要求からクライアント情報を収集し、定義済みのすべての Enterprise Gateway ルールと照合して要求を評価します。次に、Enterprise Gateway Server はルールに違反していない要求を内部サーバに渡します (2)。内部サーバは要求を処理し、応答を Enterprise Gateway Server に送信します (3)。次に、Enterprise Gateway Server はその応答をクライアントに戻します (4)。

Enterprise Gateway ポート

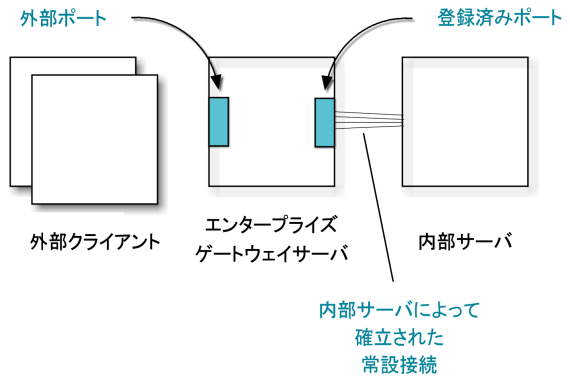
Enterprise Gateway Server として機能する Integration Server は、Enterprise Gateway外部ポートを使用して外部クライアントからの要求を受信待機します。また、Enterprise Gateway登録ポートを使用して内部サーバとの接続を維持します。セキュリティ上の理由から、内部サーバは送信接続を登録ポートに対して開始します。

注:
webMethods API Gateway が Integration Server にインストールされている場合、Enterprise Gateway外部ポートおよび Enterprise Gateway登録ポートを設定できません。この場合、Enterprise Gateway Server および Internal Server のポートは無効になります。

内部サーバによって確立されたものだけに接続を制限することで、DMZ 内のシステムが破られた場合でも、Enterprise Gateway は攻撃者が内部ネットワークに直接侵入することを困難にします。ただし、他のセキュリティメカニズムと同様に、絶対に安全というわけではありません。情報はファイアウォール内部から確立された接続を通じて DMZ から内部ネットワークに流れます。

重要:
Enterprise Gateway 設定を最大限に活用するために、Software AG では、すべての受信接続を拒否するように内部ファイアウォールを設定することを強くお勧めします。この設定を使用して、内部サーバを DMZ から隔離します。従来のサードパーティ製プロキシサーバと比較して、Enterprise Gateway Server を使用する主な利点となるのがこの機能です。

次の図は、Enterprise Gateway 設定における Enterprise Gateway の外部ポートおよび登録ポートの場所を示しています。



Enterprise Gateway 外部ポートが有効化されている Integration Server のみが、Enterprise Gateway Server と見なされます。デフォルトでは、このポートは無効化され、Enterprise Gateway Server で必要とされる一部の基本サービスを除くすべてのサービスを拒否するように設定されます。

Enterprise Gateway 外部ポートおよび登録ポートはペアで動作します。一方のポートが設定されていなければ、もう一方のポートも機能しません。

Enterprise Gateway ポートを設定する手順については、607 ページの「Enterprise Gateway ポートの設定」を参照してください。

Enterprise Gateway のルールと警告

Enterprise Gateway ルールを設定して、Enterprise Gateway Server が内部サーバに送信する要求をフィルタできます。要求がルールに違反した場合に警告を送信するように Enterprise Gateway Server を設定することもできます。

ルールが警告を送信するように設定されている場合、違反が発生すると、Enterprise Gateway Server によってサーバログに詳細が記録され、警告が生成されます。警告メッセージには、要求の送信元の IP アドレス、ユーザ情報、一致したルールフィルタの名前などの詳細情報が含まれています。

Enterprise Gateway ルールについて

要求がルールに違反する場合、Enterprise Gateway Server は要求を拒否するか、または要求を許可して違反に関する警告を送信できます。この動作は、次の 2 つのタイプの Enterprise Gateway ルールによって制御されます。

- **拒否ルール** Enterprise Gateway Server は、要求を拒否してアラートを送信します。サーバは、フィルタとの一致を検出するとすぐに要求の処理を停止し、ルールの他のフィルタまたはこの要求に対する他のルールを考慮しません。
- **警告ルール** Enterprise Gateway Server は、違反に関するアラートを送信し、要求の処理を続行します。ルールに複数のフィルタが含まれている場合、サーバは各フィルタをチェックし、一致を検出するたびに警告を送信します。1 つのルールの処理が完了すると、サーバは次のルールに進みます。後続のルールがない場合、または要求がいずれの拒否ルールにも違反していない場合、サーバは要求を許可します。

Enterprise Gateway Server は、ルールを [**Enterprise Gatewayルール**] ページに表示される順序で適用します。拒否ルールへの違反があると、Enterprise Gateway Server は要求の処理を停止するため、評価する順序に基づいてルールに優先順位を付けることが重要です。サーバは、拒否ルールを処理してから警告ルールを処理します。

Enterprise Gateway ルールを次のように設定できます。

- 要求を拒否し、違反に関する警告を送信します。
- 要求を許可し、違反に関する警告を送信します。
- すべての要求タイプに適用するか、SOAP、REST または INVOKE のいずれかの要求にのみ適用します。
- 特定の名前のサービスまたはリソースを使用するために要求に適用されます。
- 1 つ以上のフィルタを含めます。要求がフィルタで指定された条件と一致する場合、その要求はルールに違反しています。複数のフィルタを含むルールでは、要求がいずれかのフィルタと一致する場合、ルールに違反しています。たとえば、メッセージサイズ、OAuth トークンの存在、要求の送信元のモバイルアプリケーションおよびデバイスタイプに基づいて、要求をフィルタできます。

- フィルタを含めません。ルールにフィルタを指定しない場合にも、特定の要求タイプまたは特定のリソースの要求に適用されるルールを使用できます。たとえば、すべての SOAP 要求を拒否できます。

Enterprise Gateway ルールの定義の手順については、618 ページの「Enterprise Gateway ルールの使用」を参照してください。

Enterprise Gateway 警告について

要求がルールに違反した場合に Enterprise Gateway Server が送信する警告の以下の点について制御が可能です。

- ルール違反に対して Enterprise Gateway Server によって警告が発行されるかどうか。
- Enterprise Gateway Server によって警告が発行される頻度。サーバは、ルール違反が発生するごとに、または指定された時間間隔で警告を発行できます。

  時間間隔を指定した場合、Enterprise Gateway Server は要求が最初にルールに違反したときに警告を送信します。指定した間隔内に違反が再度発生した場合、この要求に対するそれ以降の警告は、サーバによって、間隔の終了まで待機した後、送信されます。
- 警告を送信するために Enterprise Gateway Server で使用される方法。サーバは、電子メールまたはフローサービスを使用して警告を送信できます。

  電子メール警告を送信するには、Enterprise Gateway Server として機能する Integration Server が電子メールを送信するように設定されている必要があります。[設定] > [リソース] ページの [**電子メール通知**] で、設定をチェックします。Enterprise Gateway Server が電子メールを送信するように設定されていない場合は、241 ページの「重大な問題に関するメッセージの電子メールアドレスへの送信」を参照してください。

  フローサービスを使用して警告を送信するために、Enterprise Gateway Server は pub.security.enterpriseGateway:alertSpec 仕様をフローサービスの署名として使用します。この仕様の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。
- ルールでデフォルト警告オプションが使用されるか、独自のカスタマイズされた警告オプションが使用されるか。

Enterprise Gateway 警告を設定する手順については、625 ページの「警告オプションの指定」を参照してください。

Enterprise Gateway Server と内部サーバ間のバージョンの相互運用性

バージョン 9.7 以前は、Enterprise Gateway Server および内部サーバが同じバージョンの Integration Server である必要がありました。しかし、バージョン 9.7 より、この要件は廃止されています。現在 Enterprise Gateway Server は、内部サーバとして機能する Integration Server の以前のバージョンに対して動作が可能です。たとえば、Enterprise Gateway Server バージョン 9.7 は、内部サーバとして Integration Server バージョン 9.7、9.6、および 9.5 SP1 と使用することができます。

Enterprise Gateway Server および内部サーバのバージョン間の相互運用性には、次のメリットがあります。

- 内部サーバとして機能する Integration Server をアップグレードすることなく、Enterprise Gateway Server として機能する Integration Server のみアップグレードできます。
- Enterprise Gateway Server の最新リリースで使用可能な機能を利用できます。

Enterprise Gateway Server および内部サーバのバージョンの相互運用性について以下の点に留意してください。

- Enterprise Gateway Server のバージョンは、内部サーバとして機能する Integration Server のバージョン以上である必要があります。
- 内部サーバとして機能する Integration Server のバージョンは、9.5 SP1 以降である必要があります。
- Software AG から、実行する Enterprise Gateway のバージョンの正しい webMethods Enterprise Gateway ライセンスを取得します。
- 初めて Enterprise Gateway をインストールする場合、内部サーバとして機能する Integration Server のバージョンに関係なく、Software AG Installer で利用可能なリリースのリストから実行する webMethods Enterprise Gateway の正しいリリースバージョンを選択する必要があります。たとえば、Enterprise Gateway 9.7 をインストールするには、Software AG Installer の [**リリース**] バージョンで webMethods 9.7 を選択する必要があります。
- Enterprise Gateway は、上書きインストールの手順を使用してアップグレードすることはできません。Enterprise Gateway をアップグレードするには、最初に Enterprise Gateway をホストする Integration Server をインストールしてから、ホスト Integration Server により高いバージョンの Enterprise Gateway をインストールします。たとえば、Integration Server 9.6 でホストされる Enterprise Gateway 9.6 を Enterprise Gateway 9.7 にアップグレードするには、まず Integration Server 9.6 をインストールする必要があります。次に、新しくインストールした Integration Server 9.6 に Enterprise Gateway 9.7 をインストールするために、Software AG Installer を再度実行する必要があります。
- Enterprise Gateway および内部サーバを別のディレクトリに、可能であれば異なるマシン上にインストールする必要があります。1 つの Integration Server を Enterprise Gateway Server および内部サーバの両方として機能するように設定しないでください。
- Enterprise Gateway の修正は Integration Server の修正の一部として提供されます。
- 内部サーバのバージョンが設定中の Enterprise Gateway Server と互換性がない場合、Integration Server は Enterprise Gateway Server および内部サーバで同じエラーメッセージをログに記録します。
- 内部サーバおよび Enterprise Gateway Server は、外部ポートに対して同じタイプのクライアント認証を使用する必要があります。

  Integration Server 9.5 SP1 は、クライアント認証でのパスワードダイジェストをサポートしていません。Enterprise Gateway Server 9.6 以上では、内部サーバがバージョン 9.5 SP1 である場合、ダイジェスト認証を使用することはできません。

従来のサードパーティ製プロキシサーバに対する Enterprise Gateway の利点

Enterprise Gateway 設定は、従来のサードパーティ製プロキシサーバと比較して数多くの利点を備えます。

- Enterprise Gateway は常設接続を使用します。これらの接続は、暗号化の利点をすべて持ちながら、SSL 接続の確立に伴う大きなオーバーヘッドを解消します。
- Enterprise Gateway を使用すると、すべての受信接続を拒否するように内部ファイアウォールを設定して、内部サーバを DMZ から隔離することができます。
- Enterprise Gateway は外部クライアントへの変更を必要としません。
- Enterprise Gateway ルールを定義してクライアント要求をフィルタできます。
- Enterprise Gateway Server として機能する Integration Server は、HTTP 要求と HTTPS 要求の両方を処理できます。一般にサードパーティ製のプロキシサーバは、どちらか一方しか処理できません。

サービス拒否保護について

Enterprise Gateway を使用してサービス拒否 (DoS) 攻撃を防止できます。DoS 攻撃の 1 つの形式は、クライアントがサーバの処理に干渉しようとして大量の要求をサーバに集中させた場合に発生します。Enterprise Gateway を使用すると、Enterprise Gateway Server で、指定した間隔内で受け入れる要求の数および同時に処理できる要求の数を制限できます。これらの制限を指定することによって、Enterprise Gateway Server および内部サーバを DoS 攻撃から保護できます。

すべての IP アドレスからの要求の合計数を考慮するように、または個別の IP アドレスからの要求数を考慮するように Enterprise Gateway Server を設定できます。たとえば、受信する要求の合計数を 10 秒間で 10 個の要求に制限し、個別の IP アドレスから受信する要求数を 10 秒間で 2 個の要求に制限する場合があります。サーバで制限の超過が検出されると、サーバは警告を送信します。設定に応じて、サーバはすべてのクライアントからの要求を一時的にブロックするか、特定の IP アドレスからの要求を拒否することができます。

DoS 攻撃を防ぐための Enterprise Gateway Server の設定の手順については、627 ページの「サービス拒否攻撃の防止」を参照してください。

信用のある IP アドレスについて

信用のあるサーバからの要求が拒否されないようにするには、IP アドレスのホワイトリストを設定して、これらの IP アドレスからの要求が常に許可されるように設定します。信用のある IP アドレスを指定するときは、以下の点に留意してください。

- Enterprise Gateway Server は、信用のある IP アドレスリストで IPv4 および IPv6 を含むアドレスをサポートします。
- CIDR (classless inter-domain routing) 表記を使用して IP アドレスの範囲を指定できます。IP アドレスの範囲を指定するには、スラッシュ (/) および CIDR サフィックスがそのあとに続く範囲に、最初の IP アドレスを入力します。

IPv4 アドレス範囲の例:

- `192.168.100.0/22` は、`192.168.100.0` から `192.168.103.255` までの IPv4 アドレスを示します。
- `148.20.57.0/30` は、`148.20.57.0` から `148.20.57.3` までの IPv4 アドレスを示します。

IPv6 アドレス範囲の例:

- `f000::/1` は、`f000::` から `ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff` までの IPv6 アドレスを示します。
- `2001:db8::/48` は、`2001:db8:0:0:0:0:0:0` から `2001:db8:0:ffff:ffff:ffff:ffff:ffff` までの IPv6 アドレスを示します。

- IP アドレスの範囲ではなく、個別の IP アドレスを指定することもできます。
- 複数の IP アドレスまたは CIDR 範囲を指定可能です。各 IP アドレスまたは CIDR 範囲はカンマ (,) で区切ります。

信用のある IP アドレスの指定手順については、627 ページの「サービス拒否攻撃の防止」を参照してください。

モバイルアプリケーション保護について

Enterprise Gateway を使用して、事前定義済みのモバイルプラットフォームのセットで特定のモバイルアプリケーションバージョンのアクセスを無効にすることができます。これらのバージョンに対してアクセスを無効にすることによって、すべてのユーザが最新バージョンのアプリケーションを使用し、最新のセキュリティおよび機能の更新を利用するようにします。

Enterprise Gateway アプリケーション保護を使用するには、Enterprise Gateway ルールでモバイルアプリケーションフィルタを定義します。これらのフィルタによって、要求の送信元のアプリケーション名、アプリケーションバージョン、デバイスタイプがチェックされます。この情報はモバイルアプリケーションによって要求ヘッダーで提示されます。フィルタ条件に一致する要求がモバイルアプリケーションから送信された場合は、Enterprise Gateway Server によって要求の拒否または警告の送信の設定済みアクションが実行されます。モバイルアプリケーションの使用を制御するための Enterprise Gateway Server の設定の手順については、630 ページの「モバイルアプリケーションの使用の制御」を参照してください。

モバイルデータの同期について

Enterprise Gateway の webMethods Mobile Support機能を使用して、モバイルデバイスとバックエンドアプリケーションとの間でデータを同期できます。この機能は以下の要素で構成されます。

- WmMobileSupport という名前の Integration Server パッケージ。ビジネス統合開発者がデータ同期ソリューションのサーバ側ビジネスロジックを作成するために必要なエレメントが含まれます。このパッケージは、ファイアウォールの内側に設置された内部サーバで使用されます。

- Mobile Support Client という名前のライブラリ。モバイルアプリケーション開発者がデータ同期要求を開始するために必要な API が含まれます。このライブラリは、モバイルアプリケーションが開発されるシステム上に存在します。

この機能を使用してモバイルデバイスとバックエンドアプリケーションとの間でデータを同期する方法の詳細については、『*Developing Data Synchronization Solutions with webMethods Mobile Support*』を参照してください。

SQL インジェクション対策について

SQL インジェクション攻撃の原因となる可能性がある要求をブロックするために、Enterprise Gateway の SQL インジェクション対策フィルタを使用できます。このフィルタを Enterprise Gateway で有効にすると、Integration Server は、潜在的な SQL インジェクション攻撃に関連する文字またはキーワードに特有のパターンについて各要求メッセージをチェックします。要求パラメータまたはペイロードで一致が検出された場合、Integration Server は要求のさらなる処理をブロックします。

Integration Server は、受信ペイロードを Enterprise Gateway で処理します。データベースで許可されていない文字が受信要求に含まれている場合、設定されたルールに基いて、Integration Server は要求を拒否して違反に関する警告を送信するか、あるいは要求を許可して違反に関する警告を送信します。

Integration Server は、SQL インジェクション攻撃を防ぐための 2 種類のフィルタを準備しています。

- **データベース固有の SQL インジェクション対策**有効の場合、Integration Server は指定されたデータベースおよび GET または POST 要求パラメータに基いて受信ペイロードをチェックします。パラメータが指定されていない場合、すべての入力パラメータが潜在的な SQL インジェクション攻撃についてチェックします。Integration Server は、パラメータの検証において ESAPI (OWASP エンタープライズセキュリティ API) に準拠します。

  パラメータは、HTTP クエリーおよび名前と値のペアが存在する HTTP フォームデータに対してのみ適用されます。

  たとえば、`http://localhost:1111/invoke/myjdbc.db:addUser?userid=' or '1'='1' --` という HTTP クエリーストリングでは、`userid` がパラメータです。

- **標準の SQL インジェクション対策**有効化されている場合、Integration Server は、メッセージ中に一重引用符 (')、ハッシュ記号 (#)、または二重ハイフン (--) を含む XML および SOAP ペイロードメッセージをブロックします。

  たとえば、次の XML ペイロードには、テキスト要素 (TITLE、ARTIST および COUNTRY) に無効な文字 '、# および -- がそれぞれ含まれています。

```
<CATALOG>
<CD>
  <TITLE>Albu'm name</TITLE>
  <ARTIST>John# Smith</ARTIST>
  <COUNTRY>USA--</COUNTRY>
  <YEAR>2014</YEAR>
</CD>
</CATALOG>
```

アンチウイルススキャンフィルタについて

アンチウイルススキャンフィルタを使用して、***ICAP (Internet Content Adaptation Protocol)*** 準拠のサーバと対話するように Enterprise Gateway を設定できます。ICAP サーバは、ウイルススキャンまたはコンテンツフィルタといった機能を実装するために使用できる複数のサービスをホストできます。アンチウイルススキャンフィルタを使用すると、Enterprise Gateway Server はすべての受信 HTTP 要求およびペイロードをウイルススキャンするために ICAP プロトコルを利用できます。

注:
アンチウイルススキャンフィルタ機能は、ICAP サーバを実装する c-icap サーバで認定され、ICAP に準拠したすべてのウイルススキャンアプリケーションと統合することができます。

アンチウイルススキャンフィルタを有効にする前に、以下の前提条件を満たしていることを確認してください。

- ICAP 準拠のサーバが DMZ にインストールおよび設定され、Enterprise Gateway Server が ICAP 準拠のサーバにアクセスできる。
- ICAP 準拠のサーバに登録済みの ICAP サービスが存在し、次の形式を使用してサービスにアクセスできる。

  ```
  icap://<icap_server>:<icap_port>/serviceName
  ```

- ICAP サーバで設定や接続の問題が発生した場合に警告を送信できるように、Enterprise Gateway Server が電子メールを送信するよう設定されている。電子メール警告が、[設定] > [リソース] ページの [**内部用電子メール**] フィールドで指定した管理者の電子メールアドレスに送信されます。

アンチウイルススキャンフィルタが Enterprise Gateway ルールの一部として有効化されている場合、Enterprise Gateway Server は次の手順で ICAP サーバの機能を使用してすべての受信ペイロードを検証します。

1. Enterprise Gateway Server は、ICAP サーバが要求をスキャンするよう要求します。
2. ICAP サーバからの応答がプレビューヘッダーを含む場合、Enterprise Gateway Server は以下の手順を実行します。
   a. Enterprise Gateway Server は、ICAP サーバから受信したプレビューヘッダーに指定されたデータの量 (バイト) で応答します。
   b. ICAP サーバが登録された ICAP サービスを使用してプレビューコンテンツをスキャンします。
   c. ICAP サーバが要求で悪意のあるコンテンツを検出すると、設定された Enterprise Gateway ルールに従って、 Enterprise Gateway Server は要求を拒否し、ルールの違反について警告を送信します。それ以外の場合、Enterprise Gateway Server は ICAP サーバがサーバ応答に基づいてスキャンするために残りのファイルを送信します。
3. ICAP サーバからの応答がプレビューヘッダーを含まない場合、Enterprise Gateway Server は以下の手順を実行します。

a. Enterprise Gateway Server は、登録された ICAP サービスを使用して要求全体をスキャンするよう ICAP サーバに要求します。

b. ICAP サーバが要求で悪意のあるコンテンツを検出すると、設定された Enterprise Gateway ルールに従って、 Enterprise Gateway Server は要求を拒否または許可し、ルールの違反について警告を送信します。

メモ:

- Enterprise Gateway は、OPTIONS メソッドへの ICAP サーバ応答に基づいて、REQMOD と RESPMOD の両メソッドをサポートします。
- Enterprise Gateway Server はすべての送信 ICAP 要求で X-wMUUID というヘッダーを送信します。このヘッダーの値は一意の識別子であり、個々のスキャンを区別するために使用できます。
- Enterprise Gateway Server では、ICAP サーバのステータスコードが 200 であり、HTTP ステータスコードが 200～300 の範囲内にある場合に成功を返します。Enterprise Gateway Server では、ICAP サーバのステータスコードが 300 以上である場合に失敗を返します。

カスタムフィルタについて

カスタムフィルタを使用して、Enterprise Gateway Server で利用できるサービスを呼び出せます。

この機能を使用して、DMZ の外部クライアントのカスタム認証、DMZのログや監査、または様々なペイロードを処理するカスタムルールの実装などのアクションを実行する、Enterprise Gateway Server のサービスのカスタマイズや呼び出しができます。

カスタムサービス実装を使用して、要求から HTTP ヘッダーとペイロードを抽出し、ビジネス要件により処理します。ヘッダーの処理において、要求の内部サーバへの転送、または要求の拒否とユーザへのエラーメッセージの返信を選択できます。

次の例について考えます。Enterprise Gateway Server へのクライアント要求に要求 URL の host:port 部分しか含まれていない場合、デフォルトで内部サーバのログオンページが表示されます。host:port 部分のみ含む要求 URL による内部サーバへのアクセスを避けるには、そのような要求を拒否するカスタムフィルタを持つ Enterprise Gateway ルールを作成できます。

また、pub.flow:setResponseHeaders および pub.flow:setResponseCode サービスも使用でき、応答にカスタムヘッダーを追加しカスタマイズされた応答コードを設定できます。

カスタムフィルタを使用して Enterprise Gateway ルールを作成する場合は、以下の点に留意してください。

- サービスにおけるカスタムロジックがその処理を必要とする場合のみ、Integration Server は Enterprise Gateway で受信したペイロードを処理します。
- Integration Server はデフォルトで要求の受信ペイロードを複製し、要求全体を処理します。Enterprise Gateway ルールを作成する場合、[**カスタムフィルタ**] の下の [**ペイロードをクローン**] チェックボックスをオフにし、Integration Server が受信ペイロードを複製せずにヘッダー情報のみ処理できるようにします。

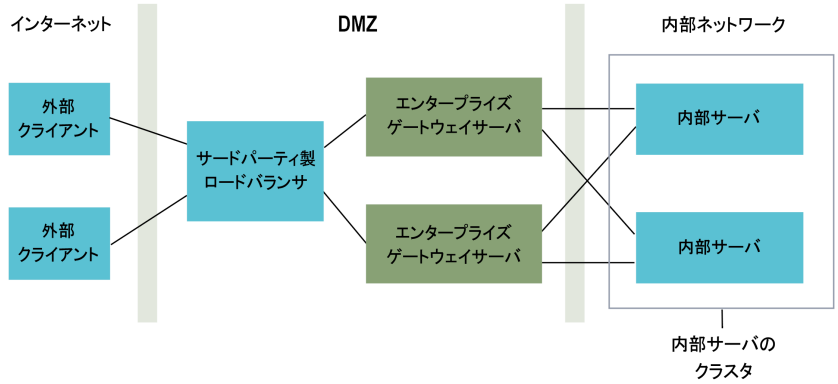
- pub.security.enterprisegateway:customFilterSpec 仕様をカスタムサービスの署名として使用します。この仕様の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。
- カスタムフィルタは、Enterprise Gateway ルールの処理中に、Enterprise Gateway Server がチェックする最後のフィルタです。
- 有効なカスタムフィルタを含む Enterprise Gateway ルールは、拒否ルールでなければなりません。カスタムフィルタを有効にすると、Enterprise Gateway Server は自動的に警告ルールを拒否ルールに変換します。

Enterprise Gateway 設定内のクラスタリング

複数の Enterprise Gateway Server を設定し、サードパーティ製品を使用して負荷分散を図ることができます。さらに、内部サーバのクラスタリングを行い、可用性、信頼性、スケーラビリティを向上させることも可能です。Integration Server クラスタリングの詳細については、*webMethods Integration Server Clustering Guide*を参照してください。

次の図は、サポートされている設定を示しています。クラスタ化された内部サーバを使用する際は、以下の点を考慮してください。

- 各内部サーバは、すべての Enterprise Gateway Server に接続する必要があります。
- Enterprise Gateway Server のクラスタを作成したり、それらを「通常の」 Integration Server のクラスタのメンバーとして加えたりしないでください。



Enterprise Gatewayの設定

このセクションでは、Enterprise Gateway を設定するための手順を説明します。次のチェックリストは、これらの手順を要約したものです。

☒	タスク	メモ
	Enterprise Gateway Server として使用する Integration Server を DMZ 内にインストールする	Integration Server を Enterprise Gateway Server に指定する場合は、インターネット上のすべての外部クライアントがこのサーバにアクセスできることに留意してください。したがって、使用可能にするサービスと定義する

☒	タスク	メモ
		ユーザのセキュリティ保護には注意する必要があります。 逆 HTTP ゲートウェイサーバ上では、開発作業およびユーザやグループの設定は行わないでください。 **重要:** 1 つの Integration Server を Enterprise Gateway Server および内部サーバの両方として機能するように設定しないでください。この設定はサポートされていません。予期できない結果を招く恐れがあります。
	Developer ユーザおよび Replicator ユーザを無効にする	これらのユーザは Enterprise Gateway Server では不要になります。これらのユーザを無効化することで、他者がこれらのユーザを通じて Enterprise Gateway Server にアクセスすることを防止できます。詳細については、115 ページの「ユーザアカウントの無効化と有効化」を参照してください。
	Enterprise Gateway 外部ポートを設定する	手順については、『607 ページの「Enterprise Gateway ポートの設定」』を参照してください。 **注:** HTTPS ポートを使用する場合は、サーバ認証、サーバの秘密鍵および認証局の認証をこのサーバに保存する必要があります。手順については、『483 ページの「Integration Server との通信のセキュリティの設定」』を参照してください。
	Enterprise Gateway 登録ポートを設定する	手順については、『607 ページの「Enterprise Gateway ポートの設定」』を参照してください。 内部サーバと Enterprise Gateway Server との間に暗号化された接続を設定する場合は、内部サーバの Administrator ユーザに対する認証を Enterprise Gateway Server に保存することもできます。詳細については、553 ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。 オプション (強く推奨)。内部サーバのみが Enterprise Gateway Server に接続できるように、IP アドレスのフィルタ処理を登録ポートに設定します。この手順により、ファイアウォールおよびユーザ認証で行われる IP アドレスのフィルタ処理を補完する、新たな保護レイヤが作成されます。 **注:**

☒	タスク	メモ
		外部ファイアウォールが Enterprise Gateway 登録ポートへの接続をフィルタで除外する場合も、内部ユーザによる Enterprise Gateway Server への接続を防止できるため、IP アドレスのフィルタは有効な方法です。 詳細については、519 ページの「ポートに接続可能な IP アドレスの制限」を参照してください。
	内部サーバを Enterprise Gateway Server に接続する	手順については、『612 ページの「内部サーバの Enterprise Gateway Server への接続 」』を参照してください。
	Enterprise Gateway および内部サーバのサーバ設定プロパティの値の設定	次のサーバ設定プロパティの値を設定するか、そのプロパティのデフォルト値が現在の状況に適しているかどうかを確認します。 ■ watt.server.rg.internalresistration.timeout ■ watt.server.rg.internalsocket.timeout ■ watt.net.socketpool.sweeperInterval

Enterprise Gateway ポートの設定

Enterprise Gateway 外部ポートおよび登録ポートはペアで動作します。一方のポートが設定されていなければ、もう一方のポートも機能しません。これらのポートを Enterprise Gateway Server 上で設定するには、以下の手順に従います。

Enterprise Gateway ポートを設定するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**設定するポートタイプの選択**] で、[**Enterprise Gatewayサーバ**] を選択します。
5. [**サブミット**] をクリックします。
6. [**Enterprise Gateway サーバ設定の編集**] ページの [**Enterprise Gateway外部ポート**] で、次の情報を入力します。

パラメータ	指定する値
[有効]	このポートを有効にするか無効にするか。ポートを無効にする場合は、後で [**ポート**] ページで有効にすることができます。
[プロトコル]	このポートで使用するプロトコル (HTTP または HTTPS)。HTTPS を選択した場合は、セキュリティおよびクレデンシャルに関する追加のボックスがページ下部に表示されます。
[ポート]	外部ポートとして使用する番号。まだ使われていない番号を使用します。これは、外部ファイアウォールを通じてクライアントが接続されるポートです。
[エイリアス]	ポートのエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。同じパッケージ名を外部ポートと登録ポートの両方に対して指定する必要があります。通常、Enterprise Gateway Server でパッケージを使用する必要はありません。したがって、デフォルト設定のままにすることができます。
[**バインドアドレス** (オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Enterprise Gateway Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 200 です。最大値は 65535 です。 **注:** [バックログ] パラメータは無効化されたポートには適用されません。Enterprise Gateway Server は、無効化されたポートへ送信する要求を拒否します。
[キープアライブタイムアウト]	アイドル状態のクライアント接続を終了するまでの時間。デフォルトは 20000 ミリ秒です。
[スレッドプール]	このポートのプライベートスレッドプールを作成するか共通スレッドプールを使用するか。 ■ サーバでこのポートについてサーバの共通スレッドプールを使用するには、[**無効**] を選択します。 ■ サーバでこのポートについてプライベートスレッドプールを作成して、他のサーバ機能とスレッドを競合しなくて済むようするには、[**有効**] を選択します。

パラメータ	指定する値
	[**スレッドプール**] が有効である場合、以下の追加パラメータも指定します。 **[スレッドプールの最小値]** Enterprise Gateway Server がサーバスレッドプールで保持するスレッドの最小数。サーバ起動時のスレッドプールには、初期値としてここで指定された最小スレッド数が設定されます。最大スレッド数に達するまで、サーバは必要に応じてスレッドをプールに追加します。デフォルトは 1 です。 **[スレッドプールの最大値]** サーバスレッドプールで保持されるスレッドの最大数。この最大スレッド数に達した場合、サーバは、現在のサービスが完了してスレッドがプールに戻されるまで待機してから、次のサービスを実行します。デフォルトは 5 です。 **[スレッドプールの優先順位]** JVM がこのスレッドプールのスレッドに割り当てる優先順位。数が大きいほど、優先順位が高くなります。デフォルトは 5 です。 **重要:** スレッドプールの優先順位はサーバのパフォーマンスとスループットに影響する可能性があるため、この設定は慎重に行ってください。 後でポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数が表示されます。

7. [**Enterprise Gateway登録ポート**] で、次の情報を入力します。

パラメータ	指定する値
[有効]	このポートを有効にするか無効にするか。ポートを無効にする場合は、後で [**ポート**] ページで有効にすることができます。
[プロトコル]	このポートで使用するプロトコル (HTTP または HTTPS)。HTTPS を選択した場合は、セキュリティおよびクレデンシャルに関する追加のボックスがページ下部に表示されます。
[ポート]	登録ポートとして使用する番号。まだ使われていない番号を使用します。 80 (HTTP の標準ポート) または 443 (HTTPS の標準ポート) などの標準ポートは使用しないようにします。これは、外部ファイアウォールが、外部からこれらの標準ポートへのアクセスを許可するためです。

パラメータ	指定する値
[エイリアス]	ポートのエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。同じパッケージ名を外部ポートと登録ポートの両方に対して指定する必要があります。通常、Enterprise Gateway Server でパッケージを使用する必要はありません。したがって、デフォルト設定のままにすることができます。
[**バインドアドレス** (オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。

8. 外部ポートまたは登録ポートの両方で、実行するクライアント認証のタイプを [**セキュリティ設定**] パネルで指定します。

   外部ポートの場合は、この設定により、ポートで受信した外部クライアントからの要求に対して実行する認証のタイプを指定します。登録ポートの場合は、この設定により、内部サーバが Enterprise Gateway Server と常設接続を確立したときに実行する認証のタイプを指定します。登録ポートに指定した設定により、Enterprise Gateway Server が内部サーバに認証を提示するよう要求するかどうかが制御されます。クライアントの認証方法の詳細については、549 ページの「クライアントの認証」を参照してください。

   注:
   デフォルトの Enterprise Gateway 設定では、Enterprise Gateway Server はクライアント認証を実行しません。正確には、サーバは認証情報 (ユーザ/パスワードまたは認証) を外部クライアントから取得し、この情報を認証のために内部サーバに渡します。ただし、Enterprise Gateway Server でもクライアント認証を実行できます。詳細については、617 ページの「Enterprise Gateway Server でのクライアント認証の実行 」を参照してください。

   以下のオプションのいずれか 1 つを選択します。

オプション	説明
[ユーザ名/パスワード]	Enterprise Gateway Server はクライアント認証を要求しません。 ■ 外部ポートの場合、サーバは、外部クライアントから送信された要求ヘッダーで、ユーザおよびパスワードについての情報を探します。 ■ 登録ポートの場合、サーバは、内部サーバからの、ユーザおよびパスワードについての情報を探します。

オプション	説明
[Digest]	外部ポートの場合、Enterprise Gateway Server はパスワードダイジェスト認証を使用します。 Enterprise Gateway Server は、外部クライアントから送信された要求ヘッダーで、パスワードダイジェストについての情報を探します。
[クライアント認証を要求する]	Enterprise Gateway Server はクライアント認証を要求します。 ■ 外部ポートの場合、サーバは、このポートを通じて受信した要求についてクライアント認証を要求します。クライアントから認証が提示されない場合は、要求ヘッダーに含まれたユーザおよびパスワードの情報を使用して要求が処理されます。 ■ 登録ポートの場合、サーバは内部サーバにクライアント認証を要求します。内部サーバが認証を提示しない場合は、ユーザおよびパスワードの情報を使用して要求が処理されます。
[クライアント認証を必須にする]	Enterprise Gateway Server はクライアント認証を必要とします。 ■ 外部ポートの場合、Enterprise Gateway Server は、このポートを通じて受信したすべての要求について、クライアント認証を必要とします。クライアントが認証を提示しなければ、要求は失敗します。 **重要:** ここでは、内部サーバに対して使用するものと同じ認証モードを使用してください。たとえば、内部サーバで認証モードを**必須**に指定するとします。Enterprise Gateway 外部ポートで**必須**と指定することで、内部サーバに渡される要求に認証が含まれるようになります。 ■ 登録ポートの場合、Enterprise Gateway Server は内部サーバのクライアント認証を必要とします。内部サーバがクライアント認証を提示しなければ、処理は失敗します。さらに、Enterprise Gateway Server 上で認証が Administrator 特権を持つユーザにマッピングされていなければ、要求は失敗します。
[ケルベロスチケットを要求する]	外部ポートの場合、Enterprise Gateway Server は外部クライアントからの要求に対してクライアント認証を必須とします。外部クライアントが認証を提示しなければ、要求は失敗します。
[ケルベロスチケットを必須にする]	外部ポートの場合、Enterprise Gateway Server は外部クライアントからのケルベロスチケットを探します。外部クライアントからチケットが提示されない場合は、要求ヘッダーに含まれたユーザおよびパスワードの情報を使用して要求が処理されます。
[JSSE の使用]	このポートで TLS 1.1、TLS 1.2、または TLS 1.3 をサポートする必要がある場合は、[**はい**] をクリックして、JSSE (Java Secure Socket

オプション	説明
	Extension) ソケットファクトリを使用するポートを作成します。この値を [**いいえ**] に設定すると、ポートはセキュアではない SSL 3.0 および TLS 1.0 のみをサポートします。デフォルトは [**はい**] です。 **注:** [**プロトコル**] フィールドで [**HTTPS**] を選択したときのみ、このフィールドは利用できます。

9. 外部ポートまたは登録ポートの [**プロトコル**] フィールドで [**HTTPS**] を選択した場合は、[**リスナー固有のクレデンシャル**] で次の情報をオプションとして入力します。

注:
[認証] ページで指定した認証とは異なる認証セットを使用する場合にのみ、この設定を使用します。

パラメータ	指定する値
[**キーストアエイリアス**]	Enterprise Gateway Server がこのポートで受信する要求に提示する認証が格納されている、キーストアの作成済みキーストアエイリアス。
[**キーエイリアス**]	指定したキーストアに格納されている特定の鍵のエイリアス。
[**トラストストアエイリアス**]	CA 署名機関に関連付けられた信用のあるルート認証が格納されているトラストストアファイルのエイリアス。

10. [**変更内容の保存**] をクリックします。

Enterprise Gateway 外部ポートおよび登録ポートの削除

Enterprise Gateway 外部ポートおよび登録ポートを初めて設定する場合は、各ポートをパッケージに関連付けます。いずれか一方のポートを削除すると、Enterprise Gateway Server はもう一方のポートも削除します。

ポートは Integration Server Administrator の [**ポート**] ページで削除します。詳細については、167 ページの「ポートの設定」を参照してください。

内部サーバの Enterprise Gateway Server への接続

ここでは、内部サーバを Enterprise Gateway Server に接続する方法について説明します。

内部サーバを Enterprise Gateway Server に接続するには

1. 内部サーバとして機能する Integration Server 上で Integration Server Administrator を開きます。

2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。
4. [**設定するポートタイプの選択**] で、[**内部サーバ**] を選択します。
5. [**サブミット**] をクリックします。
6. [**内部サーバ設定の編集**] ページの [**内部サーバ**] で、次の情報を入力します。

パラメータ	指定する値
[有効]	このポートを有効にするか無効にするか。ポートを無効にする場合は、後で [**ポート**] ページで有効にすることができます。
[プロトコル]	このポートで使用するプロトコル (HTTP または HTTPS)。HTTPS を選択した場合は、セキュリティおよびクレデンシャルに関する追加のボックスがページ下部に表示されます。
[パッケージ名]	このポートに関連付けるパッケージ。通常、内部サーバでパッケージを使用する必要はありません。したがって、デフォルト設定のままにすることができます。
[エイリアス]	ポートのエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[最大接続数]	Enterprise Gateway Server と内部サーバの間で維持される接続の数。デフォルトは 5 です。 **注:** 最適なパフォーマンスを得るには、リスナーの [最大接続数] の設定を、サーバスレッドプール設定の最大スレッド数よりも少し小さい値に設定します。サーバスレッドプールの最大スレッド数は、[設定] > [リソース] ページで設定します。内部サーバに複数のリスナーを定義する場合は、[最大接続数] の設定の合計が、サーバスレッドプールの最大スレッド数の設定よりも少し小さい値になるようにします。[最大接続数] を内部サーバのスレッドプールと等しい値には設定しないでください。スケジュール済みサービスやトリガーの実行および内部サーバに直接接続するユーザを処理できる十分な数のスレッドを確保してください。
[スレッドプール]	このポートのプライベートスレッドプールを作成するか共通スレッドプールを使用するか。 ■ サーバでこのポートについてサーバの共通スレッドプールを使用するには、[**無効**] を選択します。

パラメータ	指定する値
	■ サーバでこのポートについてプライベートスレッドプールを作成して、他のサーバ機能とスレッドを競合しなくて済むようするには、[**有効**] を選択します。 [**スレッドプール**] が有効である場合、以下の追加パラメータも指定します。 **[スレッドプールの最小値]** サーバスレッドプールで保持されるスレッドの最小数。サーバ起動時のスレッドプールには、初期値としてここで指定された最小スレッド数が設定されます。最大スレッド数に達するまで、サーバは必要に応じてスレッドをプールに追加します。デフォルトは 1 です。 **[スレッドプールの最大値]** サーバスレッドプールで保持されるスレッドの最大数。この最大スレッド数に達した場合、サーバは、現在のサービスが完了してスレッドがプールに戻されるまで待機してから、次のサービスを実行します。デフォルトは 5 です。 **[スレッドプールの優先順位]** JVM がこのスレッドプールのスレッドに割り当てる優先順位。数が大きいほど、優先順位が高くなります。デフォルトは 5 です。 **重要:** スレッドプールの優先順位はサーバのパフォーマンスとスループットに影響する可能性があるため、この設定は慎重に行ってください。 後でポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数が表示されます。

7. [**Enterprise Gateway サーバ**] で、以下の情報を入力します。

パラメータ	指定する値
[ホスト]	Enterprise Gateway Server が稼動しているマシンのホスト名または IP アドレス。
[ポート]	Enterprise Gateway Server 上の登録ポートのポート番号。

8. [**プロトコル**] ボックスで [**HTTPS**] を選択した場合は、[**登録クレデンシャル**] で次の情報をオプションとして入力します。ここで指定する登録クレデンシャルは、Enterprise Gateway 登録ポートの設定内容と一致する必要があります。

パラメータ	指定する値
[ユーザ名]	内部サーバの接続に使用される Enterprise Gateway Server のユーザ名。
[パスワード]	内部サーバの接続に使用される Enterprise Gateway Server のユーザのパスワード。
[JSSE の使用]	このポートで TLS 1.1、TLS 1.2、または TLS 1.3 をサポートする必要がある場合は、[**はい**] をクリックして、JSSE (Java Secure Socket Extension) ソケットファクトリを使用するポートを作成します。この値を [**いいえ**] に設定すると、ポートはセキュアではない SSL 3.0 および TLS 1.0 のみをサポートします。デフォルトは [**はい**] です。 **注:** [**プロトコル**] フィールドで [**HTTPS**] を選択したときのみ、このフィールドは利用できます。
[キーストアエイリアス]	内部サーバがクライアント認証のために Enterprise Gateway Server に送信する認証が格納されているキーストアの作成済みキーストアエイリアス。内部サーバは、Enterprise Gateway Server への最初の登録接続を確立する際に、この認証を送信します。内部サーバは、Enterprise Gateway Server より要求があったときのみこの認証を送信します。 この値は、[認証] ページで指定されている認証とは異なるサーバ認証を提示させる場合のみ指定します。
[キーエイリアス]	上記で指定したキーストアに格納されている、鍵ペアおよび関連する認証の作成済みキーエイリアス。
[トラストストアエイリアス]	CA 署名機関に関連付けられた信用のあるルート認証が格納されているトラストストアファイルのエイリアス。

9. [**外部クライアントのセキュリティ**] の [**クライアントの認証**] リストで、内部サーバが外部クライアントに対して実行するクライアント認証のタイプを指定します。外部クライアントは認証情報を Enterprise Gateway Server に渡し、その認証情報が内部サーバに渡されます。

オプション	説明
[ユーザ名/パスワード]	内部サーバは外部クライアントのクライアント認証を要求しません。代わりに、要求ヘッダーでユーザおよびパスワードについての情報を探します。
[Digest]	内部サーバは、要求ヘッダーでパスワードダイジェストについての情報を探します。
[クライアント認証を要求する]	内部サーバは外部クライアントからの要求に対してクライアント認証を要求します。外部クライアントから認証が提示されない場合

オプション	説明
	は、要求ヘッダーに含まれたユーザおよびパスワードの情報を使用して要求が処理されます。
[クライアント認証を必須にする]	内部サーバは外部クライアントからの要求に対してクライアント認証を必須とします。外部クライアントが認証を提示しなければ、要求は失敗します。
[ケルベロスチケットを要求する]	内部サーバは外部クライアントからの要求に対してクライアント認証を必須とします。外部クライアントが認証を提示しなければ、要求は失敗します。
[ケルベロスチケットを必須にする]	内部サーバは外部クライアントからのケルベロスチケットを探します。外部クライアントからチケットが提示されない場合は、要求ヘッダーに含まれたユーザおよびパスワードの情報を使用して要求が処理されます。

重要:
ここでは、Enterprise Gateway 外部ポートで使用した認証モードと同じモードを使用してください。たとえば、内部サーバと Enterprise Gateway 外部ポートの両方で **[クライアント認証を必須にする]** を指定することで、内部サーバに渡される要求に認証が含まれるようになります。

クライアント認証の処理の詳細については、549 ページの「クライアントの認証」を参照してください。

10. **[変更内容の保存]** をクリックします。

Enterprise Gateway 登録ポートへの接続の表示

単一の Enterprise Gateway 登録ポートについて、内部サーバ接続のリストを表示できます。

Enterprise Gateway 登録ポートへの接続を表示するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. **[サーバ]** > [ポート] に移動します。
3. Enterprise Gateway 登録ポートのポート番号をクリックします。
4. **[サーバ]** > [ポート] > [エンタープライズゲートウェイサーバの詳細の表示] ページで、**[Enterprise Gateway登録ポートへの接続の表示]** をクリックします。

   Integration Server Administrator に **[サーバ]** > [ポート] > [Enterprise Gateway 登録ポート接続] ページが表示されます。[Enterprise Gateway 登録ポート接続: *ポート番号*] 表に、指定さ

れた *ポート番号* の Enterprise Gateway 登録ポートへの内部サーバ接続がリストされます。Integration Server Administrator には、接続ごとに以下の情報が表示されます。

フィールド	説明
[ホスト]	Enterprise Gateway 登録ポートに接続された内部サーバの IP アドレス。
[ポート]	リモート内部サーバ接続用に作成されたソケットのローカルポート番号。
[接続済み (*接続数*)]	内部サーバが Enterprise Gateway 登録ポートに接続されているかどうか。

Enterprise Gateway Serverでのクライアント認証の実行

デフォルトの Enterprise Gateway 設定では、外部クライアントが要求を Enterprise Gateway Server に送信し、そこから外部クライアントに関する認証情報 (ユーザ/パスワードまたは認証) が内部サーバに転送されます。内部サーバが認証を実行します。2つのファイアウォールの内側にある内部サーバに保存されていると、認証の安全性が高くなるため、この設定をお勧めします。

ただし、内部サーバで実行される認証のほかに、Enterprise Gateway Server でもクライアント認証を実行するように設定することも可能です。

Enterprise Gateway Server でのクライアント認証を有効にするには

1. Enterprise Gateway Server として機能する Integration Server 上の Integration Server Administrator で、[設定] > **[拡張設定]** ページに移動して、watt.server.revInvoke.proxyMapUserCerts システムプロパティを「true」に設定します。

2. 認証を要求するまたは必須とするように Enterprise Gateway Server を設定している場合、アクセスを許可する各外部クライアントについて、Enterprise Gateway Server には、ユーザにマッピングされたクライアントの公開認証のコピーが必要です。認証のマッピングの詳細については、553 ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。

   認証を要求するまたはユーザ名とパスワードを使用して認証を実行するように Enterprise Gateway Server を設定している場合は、Enterprise Gateway Server にはクライアントのユーザ名が必要です。

   インポートされた外部クライアントの認証またはユーザ名が、Enterprise Gateway Server と内部サーバの両方で同じであることを確認してください。

3. Enterprise Gateway 外部ポートのクライアント認証モードを **[クライアント認証を必須にする]** に設定します。

   a. **[サーバ]** > [ポート] に移動します。

b. Enterprise Gateway 外部ポートを示す行を探します。ポート番号をクリックして、[**HTTP ポート設定の編集**] をクリックします。

c. [Enterprise Gateway Server 設定の編集] ページの [**Enterprise Gateway外部ポート**] 領域にある [**クライアントの認証**] ボックスで、[**クライアント認証を必須にする**] を選択します。

> **注:**
> クライアント認証は、HTTPS プロトコルでのみサポートされます。[**クライアントの認証**] ボックスが表示されない場合は、外部ポートプロトコルを [**HTTPS**] に変更します。

d. [**変更内容の保存**] をクリックします。

Enterprise Gateway ルールの使用

Enterprise Gateway ルールは、Enterprise Gateway Server を通過して内部サーバへ到達できる要求を決定する 1 つ以上のフィルタで構成されます。

Enterprise Gateway ルールの作成

Enterprise Gateway ルールを作成する場合は、以下の点に留意してください。

- 電子メール警告を送信するには、Enterprise Gateway Server として機能する Integration Server が電子メール警告を送信するように設定されている必要があります。詳細については、241 ページの「重大な問題に関するメッセージの電子メールアドレスへの送信」を参照してください。
- ルールは、評価する順序に従って優先順位付けしてください。このことは、拒否ルールの場合に特に重要になります。これは、拒否ルールに違反すると、Enterprise Gateway Server によって要求の処理が停止され、場合によってはルール全体が評価されないためです。
- フィルタを含まないルールを作成できます。特定の要求タイプのすべての要求を拒否したり、特定のリソースを使用する要求に関して警告を送信したりする場合に、このようなルールを使用できます。
- 新しいルールを作成すると、Enterprise Gateway Server によって、デフォルト警告オプションがそのルールに適用されます。ルールの警告オプションをカスタマイズするには、627 ページの「ルール固有の警告オプションの指定」を参照してください。
- Enterprise Gateway ルールは、拒否ルールと警告ルールにカテゴリ分けされます。新しいルールを作成すると、そのルールは [**セキュリティ**] > [Enterprise Gateway ルール] ページで、該当するカテゴリのルールのリストの最後に追加されます。

Enterprise Gateway ルールを作成するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。

2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。

3. [**ルールの作成**] をクリックします。

4. [**ルール名**] ボックスに、ルールの名前を入力します。

   有効なルール名の条件:

   - 一意であること。
   - NULL 以外であること。
   - 空白を使用していないこと。
   - 以下の特殊文字を使用していないこと。

     ?~ ` !@ # $ % ^ & * ( ) - + = { } | [ ] \\ : \" ; ' < > , /

5. [**説明**] ボックスに、ルールの簡単な説明を入力します。

6. ルールを直ちに有効にする場合は、[**有効**] チェックボックスをオンにします。ルールは有効な場合にのみ、Enterprise Gateway Server によって要求に適用されます。後で [Enterprise Gatewayルール] ページからルールを有効にすることができます。

7. [**要求タイプ**] リストから、Enterprise Gateway ルールを適用する要求のタイプを選択します。このルールをすべての要求に適用する場合は、[**すべて**] を選択します。このルールを SOAP、REST または INVOKE 要求に適用するには、それぞれ [**SOAP**]、[**REST**] または [**呼び出し**] を選択します。

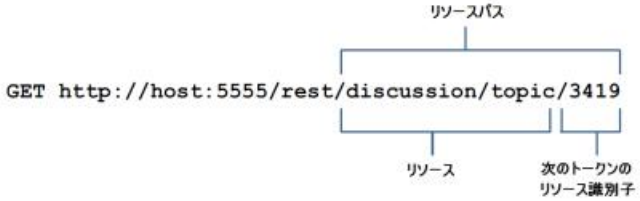
8. 要求されるリソースに基づいて要求をフィルタするには、[**リソースパス**] ボックスでストリングを *folder_name/service_name* の形式で指定します。複数のパスを 1 行に 1 つずつ指定でき、アスタリスク (*) のワイルドカード文字を指定できます。このボックスは、[**要求タイプ**] リストから要求タイプを選択した場合にのみ使用できます。

   次に、INVOKE、Web サービスおよび REST の各要求のリソースパスの例をいくつか示します。

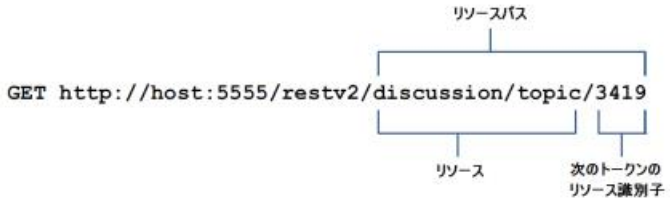
   INVOKE 要求の例

   Web サービス要求の例

REST 要求 - rest ディレクティブの例



REST 要求 - restv2 ディレクティブの例



9. [**アクション**] ボックスで、以下のいずれかを選択します。

選択項目	目的
[要求の拒否と警告]	ルールに違反する場合は要求を拒否します。設定済み警告オプションに基づいて、Enterprise Gateway Server によって警告が送信されます。
[警告]	要求を許可し、設定済み警告オプションに基づいて警告を送信します。

10. [**エラーメッセージ**] ボックスで、要求が Enterprise Gateway 拒否ルールに違反した場合に Enterprise Gateway Server によってクライアントに送信するカスタムメッセージを入力します。サーバは、このメッセージを HTTP 403 ステータスコードの状態メッセージとして送信します。このステータスコードの形式は、「HTTP 403 *custom error message*」です。

11. Enterprise Gateway Server で要求の認証ヘッダーの OAuth トークンをチェックする場合は、[**OAuth フィルタ**] で [**OAuth クレデンシャルが必要**] チェックボックスをオンにします。このフィルタをすぐに有効にするには、[**有効**] チェックボックスをオンにします。このフィルタを後で有効にする場合は、ルールの編集ページでルールを編集します。

12. [**メッセージサイズフィルタ**] の [**最大メッセージサイズ**] ボックスで、HTTP および HTTPS 要求で許可される最大サイズをメガバイト単位で入力します。この制限で指定したサイズよ

りも要求が大きい場合は、要求はルール違反となり、Enterprise Gateway Server によって設定済みアクションが実行されます。このフィルタをすぐに有効にするには、[**有効**] チェックボックスをオンにします。このフィルタを後で有効にする場合は、ルールの編集ページでルールを編集します。

13. [**モバイルアプリケーションの保護フィルタ**] で、要求の送信元のデバイスタイプおよび要求を送信したモバイルアプリケーションのバージョンに基づいて要求を説明する 1 つ以上の条件を指定します。要求が条件を満たす場合は、ルール違反となり、Enterprise Gateway Server によって設定済みアクションが実行されます。たとえば、ZZZ スマートフォンから送信された ABC アプリケーションの古いバージョンからの要求を許可しないように指定できます。このフィルタをすぐに有効にするには、[**有効**] チェックボックスをオンにします。このフィルタを後で有効にする場合は、ルールの編集ページでルールを編集します。

14. [**SQL インジェクション対策フィルタ**] で、次のいずれかまたは両方を選択します。

選択項目	目的
[**データベース固有の SQL インジェクション対策**]	特定のデータベースに対して、潜在的な SQL インジェクション攻撃と認識されるパターンを含む受信要求をブロックします。 ■ [**データベース**] リストから、特定のパラメータをチェックする必要があるデータベースを選択します。 ■ [**パラメータ**] で、受信要求に存在する可能性のある 1 つ以上の GET または POST 要求パラメータを指定します。 [**パラメータ**] には、英数字、ドル記号 ($)、アンダースコア (_) のみ使用できます。
[**標準の SQL インジェクション対策**]	メッセージ中に一重引用符 (')、番号記号 (#)、または二重ハイフン (--) を含む XML または SOAP ペイロードメッセージをブロックします。

注:
フィルタは両方とも選択できますが、複数のオプションを選択することで、各フィルタで要求メッセージの検査を個別に行う必要があるため、メッセージの処理時間が増加します。

このフィルタをすぐに有効にするには、[**有効**] チェックボックスをオンにします。このフィルタを後で有効にする場合は、ルールの編集ページでルールを編集します。

15. Enterprise Gateway Server で受信ペイロードのウィルススキャンを行うには、[**アンチウイルススキャンフィルタ**] で次の情報を入力します。

パラメータ	指定する値
[**アンチウイルス ICAP エンジン名**]	ICAP サーバの名前。

パラメータ	指定する値
[ICAP ホスト名または IP アドレス]	ICAP サーバが稼動しているマシンのホスト名または IP アドレス。
[ICAP ポート番号]	ICAP サーバが受信待機するポート番号。
[ICAP サービス名]	ペイロードをウイルススキャンするために使用できる、ICAP サーバによって表示されるサービスの名前。

このフィルタをすぐに有効にするには、[**有効**] チェックボックスをオンにします。このフィルタを後で有効にする場合は、ルールの編集ページでルールを編集します。

16. Enterprise Gateway Server で利用できるサービスを呼び出すには、[**カスタムフィルタ**] で、[**呼び出しサービス**] フィールドの隣にある [**参照**] ボタンをクリックします。[**パッケージ名**] リストから、呼び出したいサービスを含むパッケージを選択します。Enterprise Gateway Server は [**サービス名**] リストに、選択したパッケージの署名として pub.security.enterprisegateway:customFilterSpec 仕様をもつサービス名を入力します。呼び出したいサービスを選択します。

    [**実行ユーザ**] ボックスで、をクリックして Enterprise Gateway Server がサービスを実行するときに使用するユーザ名を入力します。デフォルトは Administrator です。

    ローカルディレクトリまたはセントラルディレクトリからユーザを選択できます。Enterprise Gateway Server は、指定したユーザがサービスを呼び出した認証ユーザであると見なして、サービスを実行します。サービスが ACL で管理されている場合は、必ずサービスの呼び出しを許可されているユーザを指定してください。

    Integration Server が受信ペイロードを複製せず、要求のヘッダー情報のみを処理するよう、[**ペイロードをクローン**] チェックボックスをオフにします。チェックボックスをオンにすると、Integration Server は受信ペイロードを複製し要求全体を処理します。デフォルトで、チェックボックスはオンになっています。

    重要:
    カスタムフィールドが有効な場合は、カスタムフィルタを含む Enterprise Gateway ルールは拒否ルールでなければなりません。カスタムフィルタを有効にすると、Enterprise Gateway Server は自動的に警告ルールを拒否ルールに変換します。

    このフィルタをすぐに有効にするには、[**有効**] チェックボックスをオンにします。このフィルタを後で有効にする場合は、ルールの編集ページでルールを編集します。

17. [**変更内容の保存**] をクリックします。

    Enterprise Gateway Server により、[Enterprise Gatewayルール] ページの [**拒否ルール**] または [**警告ルール**] にルールが表示されます。どちらに表示されるかは、ルールの作成時に選択したアクションで決まります。

Enterprise Gateway ルールの有効化

ルールは有効な場合にのみ、Enterprise Gateway Server によって要求に適用されます。ルールは作成時に有効化できますが、この手順によって後で有効化することもできます。

Enterprise Gateway ルールを有効化するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [Enterprise Gateway ルール] ページで、有効にするルールの [**有効**] 列の [**いいえ**] をクリックします。
4. ルールを有効にすることを求めるプロンプトへの応答で、[**OK**] をクリックします。

   ルールが有効になると、[**有効**] 列に ✓ アイコンと [**はい**] が表示されます。

Enterprise Gateway ルールの無効化

特定の Enterprise Gateway ルールを、削除することなく、要求に適用されないようにする場合は、無効にすることができます。

注:
ルールを無効にしても、[Enterprise Gateway ルール] ページ上の優先順位には影響しません。

Enterprise Gateway ルールを無効化するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [Enterprise Gateway ルール] ページで、無効にするルールの [**有効**] 列の [**はい**] をクリックします。
4. ルールを無効にすることを求めるプロンプトへの応答で、[**OK**] をクリックします。

   ルールが無効になると、[**有効**] 列に [**いいえ**] が表示されます。

Enterprise Gateway ルールの編集

Enterprise Gateway ルールの編集では、名前以外のすべてのプロパティを編集できます。拒否ルールを警告ルールに (またはその逆に) 変更した場合、Enterprise Gateway Server は変更され

たルールを対応する拒否ルールまたは警告ルールのリストの最下部に追加します。したがって、そのカテゴリで最も低い優先順位がそのルールに割り当てられます。ルールの優先順位を変更するには、624 ページの「Enterprise Gateway ルールの優先順位の変更」を参照してください。

Enterprise Gateway ルールを編集するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [Enterprise Gateway ルール] ページのルールリストで、編集するルールの名前をクリックします。
4. 選択したルールのプロパティページで、必要な変更を行います。
5. [**変更内容の保存**] をクリックします。

Enterprise Gateway ルールのコピー

既存のルールをコピーして新しい Enterprise Gateway ルールを作成できます。ルールをコピーすると、ルール名を除き、フィルタ設定、警告オプションなどのプロパティがコピーされます。

Enterprise Gateway ルールをコピーするには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [Enterprise Gateway ルール] ページで、コピーするルールの [**コピー**] 列の アイコンをクリックします。

   「_copy」がルール名に付加され、ルールのプロパティページが表示されます。
4. 必要に応じてルールのプロパティおよびフィルタを表示または変更します。詳細については、618 ページの「Enterprise Gateway ルールの作成」を参照してください。
5. [**変更内容の保存**] をクリックします。

Enterprise Gateway ルールの優先順位の変更

Enterprise Gateway Server は、Enterprise Gateway ルールを [Enterprise Gateway ルール] ページに表示される順序で適用します。たとえば、要求が拒否ルールのフィルタ条件を満たした場合、サーバによって要求は拒否され、ルールの次のフィルタまたは別のルールには進みません。ルールの優先順位を変更して、意図した順序でルールが適用されるようにすることができます。

Enterprise Gateway ルールの優先順位を変更するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [Enterprise Gateway ルール] ページの移動するルールの [**優先度**] 列で、次のいずれかの操作を行います。

目的	クリックするボタン
リスト内でルールを上に移動	⇧
リスト内でルールを下に移動	⇩

Enterprise Gateway ルールの削除

Enterprise Gateway ルールが不要になった場合は、削除できます。

Enterprise Gateway ルールを削除するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [Enterprise Gateway ルール] ページで、削除するルールに対応する行の ✕ アイコンをクリックします。
4. [**削除**] をクリックします。ルールを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

警告オプションの指定

すべてのルールに適用されるデフォルト警告オプションを指定できます。また、ルール固有の警告オプションも指定できます。

デフォルト警告オプションの指定

デフォルトでは、Enterprise Gateway ルールに警告タイプが指定されていない場合、Enterprise Gateway Server では要求がルールに違反したときに警告を送信しません。ルールに警告タイプが指定されていなくても警告を発行する場合は、デフォルト警告オプションを設定できます。

注:

ルール違反に関する電子メール警告を送信するには、Enterprise Gateway Server として機能する Integration Server が、電子メールを送信するように設定されている必要があります。[設定] > [リソース] ページの [**電子メール通知**] で、設定をチェックします。Enterprise Gateway Server が電子メールを送信するように設定されていない場合は、241 ページの「重大な問題に関するメッセージの電子メールアドレスへの送信」を参照してください。

デフォルト警告オプションを指定するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [**デフォルト警告オプション**] を選択し、[**デフォルト警告オプションの編集**] をクリックします。
4. [**警告タイプ**] リストから、次のいずれかのオプションを選択します。

選択項目	目的
[なし]	警告を送信しません。これがデフォルトです。
[電子メール]	電子メール警告を送信します。
[フローサービス]	フローサービスを呼び出してルール違反を通知します。

5. [**警告の送信**] ボックスで、以下のいずれかを実行します。
   - 要求がルールに違反するたびに Enterprise Gateway Server によって警告が送信されるようにする場合は、[**ルール違反時**] を選択します。
   - 指定した間隔で Enterprise Gateway Server によって警告が送信されるようにする場合は、[**スケジュール間隔**] を選択し、間隔 (分単位) を入力します。要求がルールに違反すると、サーバによって警告が送信されます。指定した間隔内に違反が再度発生した場合、サーバによって、間隔の終了まで待機した後、警告が送信されます。たとえば、間隔を 1 分に指定したとします。10 時に、要求がルールに違反し、サーバによって警告が送信されます。10 秒後に、別の要求がルールに違反します。10 秒後に、さらに別の要求がルールに違反します。サーバでは、10 時 1 分まで、間隔内に生成された警告の概要を送信するのを待機します。
6. 警告タイプとして電子メールを選択した場合、[**電子メール ID**] ボックスに警告の送信先の電子メールアドレスを入力します。複数の電子メールアドレスを入力できます。複数のアドレスを入力する場合は、アドレスをセミコロン (;) で区切ります。
7. 警告タイプとしてフローサービスを選択した場合、以下の操作を行います。
   a. [**フォルダ名.サブフォルダ名:サービス名**] ボックスに、Enterprise Gateway Server で実行するフローサービスの完全修飾名を入力します。

> 注:
> pub.security.enterpriseGateway:alertSpec 仕様をフローサービスの署名として使用します。この仕様の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

b. [**実行ユーザ**] ボックスに、Enterprise Gateway Server がサービスを実行するときに使用するユーザ名を入力します。アイコンをクリックしてユーザを検索して選択します。ローカルディレクトリまたはセントラルディレクトリからユーザを選択できます。

   Enterprise Gateway Server は、指定したユーザがサービスを呼び出した認証ユーザであると見なして、サービスを実行します。サービスが ACL で管理されている場合は、必ずサービスの呼び出しを許可されているユーザを指定してください。

8. [**変更内容の保存**] をクリックします。

ルール固有の警告オプションの指定

デフォルトのルール警告オプションは、すべてのルールにわたってグローバルレベルで適用されます。ルール固有の警告オプションを指定することもできます。

ルール固有の警告オプションを指定するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。

2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。

3. [Enterprise Gateway ルール] ページのルールのリストの [**警告オプション**] 列は、ルールにデフォルト警告オプションが設定されているかカスタマイズされた警告オプションが設定されているかを示しています。[**デフォルト**] または [**カスタム**] をクリックして、[**警告オプションの編集**] ページを開きます。

4. 必要に応じて警告オプションを編集します。警告オプションを設定する手順については、625 ページの「デフォルト警告オプションの指定」を参照してください。

5. [**変更内容の保存**] をクリックします。

サービス拒否攻撃の防止

Enterprise Gateway Server を設定して、サーバが受け入れる要求数およびサーバで同時に処理する要求数を制限できます。このように制限することで、大量の受信要求を使用してサーバの処理に干渉するサービス拒否攻撃から Enterprise Gateway Server および内部サーバの両方を保護できます。

Enterprise Gateway Server で要求をグローバルに制限するか、IP アドレスによって制限するか、あるいはその両方で制限するかを設定できます。

ただし、信用のあるサーバからの要求が拒否されないようにするために、信用のある IP アドレスのリストを指定できます。これらの信用のある IP アドレスからの要求は常に許可されます。信用のある IP アドレス範囲の詳細については、600 ページの「信用のある IP アドレスについて」を参照してください。

要求のグローバルな制限

Enterprise Gateway Server で受け入れる要求の合計数および同時に処理する数を制限するには、以下の手順に従います。グローバルオプションを選択すると、送信元の IP アドレスに関係なく、サーバによってすべての受信要求が考慮されます。信用のある IP アドレスのリストを設定すると、これらの IP アドレスからの要求は要求制限で考慮されません。要求数が設定した制限を超えると、指定した時間、サーバによってすべての要求がすべての Enterprise Gateway 外部ポートでブロックされます。ただし、Enterprise Gateway Server は設定された制限に到達した後も信用のある IP アドレスからの要求を受信し続けます。

Enterprise Gateway Server で受け入れる個別の IP アドレスからの要求数を制限する場合は、629 ページの「IP アドレスによる要求の制限」を参照してください。

グローバルと IP アドレスの両方のオプションを指定した場合は、Enterprise Gateway Server によってグローバル処理が最初に実行されます。

要求をグローバルに制限するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [**サービスの拒否オプション**] をクリックし、次に [**グローバルにサービスを拒否の設定**] をクリックします。
4. [**有効**] チェックボックスをオンにします。
5. [**要求の最大数**] ボックスに、Enterprise Gateway Server で特定の間隔に受け入れる要求の最大数を入力します。次に、時間間隔を秒単位で入力します。
6. [**一度に処理する要求の最大数**] ボックスに、同時に処理可能な要求の最大数を入力します。
7. [**ブロック間隔**] ボックスに、すべての Enterprise Gateway 外部ポートで Enterprise Gateway Server によって要求がブロックされる時間 (分) を入力します。
8. [**エラーメッセージ**] ボックスに、要求が拒否されたときにクライアントに送信するカスタムメッセージを必要に応じて入力します。
9. [**信用のある IP アドレス範囲**] ボックスに、信用のある IPv4 または IPv6 アドレスを入力し、これらの IP アドレスが常に許可されるようにします。複数の IP アドレスまたは IP アドレ

スの範囲はカンマ (,) で区切って指定します。信用のある IP アドレス範囲の詳細については、600 ページの「信用のある IP アドレスについて」を参照してください。

10. [**変更内容の保存**] をクリックします。

IP アドレスによる要求の制限

個別の IP アドレスから Enterprise Gateway Server が受け入れる要求数または同時に処理可能な要求数を制限するには、以下の手順に従います。

ある IP アドレスからの要求数が設定した制限を超えると、Enterprise Gateway Server によって、すべての Enterprise Gateway 外部ポートで、その IP アドレスからの要求は永続的に、または指定した時間、拒否されます。ただし、Enterprise Gateway Server は設定された制限に到達した後も信用のある IP アドレスからの要求を受信し続けます。

IP アドレスに関係なく Enterprise Gateway Server で受け入れる要求の合計数を制限する場合は、628 ページの「要求のグローバルな制限」を参照してください。

グローバルと IP アドレスの両方のオプションを指定した場合は、Enterprise Gateway Server によってグローバル処理が最初に実行されます。

IP アドレスによって要求を制限するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。
2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。
3. [**サービスの拒否オプション**] をクリックし、次に [**IP アドレスによりサービスを拒否の設定**] をクリックします。
4. [**有効**] チェックボックスをオンにします。
5. [**要求の最大数**] ボックスに、Enterprise Gateway Server で特定の間隔に特定の IP アドレスから受け入れる要求の最大数を入力します。次に、時間間隔を秒単位で入力します。
6. [**一度に処理する要求の最大数**] ボックスに、Enterprise Gateway Server で個別の IP アドレスから同時に処理可能な要求の最大数を入力します。
7. [**上限を超えたとき**] ボックスに、信用のない IP アドレスからの要求数が指定した制限を超えた場合に実行するアクションを指定します。
   - この IP アドレスからの今後の要求を永続的に拒否するには、[**拒否リストに追加**] を選択します。Enterprise Gateway Server によって IP アドレスは Enterprise Gateway 拒否リストに追加されます。そのため、この IP アドレスからの要求は、すべての Enterprise Gateway 外部ポートで拒否されます。管理者が拒否リストから削除するまで、IP アドレスはリストに残ります。

注:
この Enterprise Gateway 拒否リストは、Enterprise Gateway 外部ポートのポートレベルの許可/拒否リストよりも優先されます。

- このIPアドレスからの要求を一時的にブロックするには、[**ブロック**] を選択します。[**ブロック間隔**] ボックスに、要求をブロックする時間 (分) を指定します。要求をブロックするために、Enterprise Gateway Server によって IP アドレスは Enterprise Gateway 拒否リストに追加されます。そのため、サーバは、すべてのEnterprise Gateway 外部ポートでこの IP アドレスからの要求を設定された時間だけ拒否します。

8. [**エラーメッセージ**] ボックスに、要求が拒否されたときにクライアントに送信するカスタムメッセージを必要に応じて入力します。

9. [**信用のある IP アドレス範囲**] ボックスに、信用のある IPv4 または IPv6 の IP アドレスまたは範囲を入力し、これらの IP アドレスが常に許可されるようにします。複数の IP アドレスまたは IP アドレスの範囲はカンマ (,) で区切って指定します。信用のある IP アドレスの指定の詳細については、600 ページの「信用のある IP アドレスについて」を参照してください。

10. [**変更内容の保存**] をクリックします。

モバイルアプリケーションの使用の制御

特定のバージョンのアプリケーションおよび特定のデバイスタイプからの要求のみを許可することによって、モバイルアプリケーションからの要求を規制できます。そうすることによって、すべてのユーザが最新バージョンのモバイルアプリケーションを使用し、最新のセキュリティおよび機能の更新を利用するようにします。

モバイルアプリケーションの使用を制御するには、最初に規制するデバイスタイプのリストおよびモバイルアプリケーションのリストを定義します。次に、Enterprise Gateway ルールのモバイルアプリケーションフィルタを設定するときに、これらの値から選択します。モバイルアプリケーションからは、デバイスタイプ、アプリケーション名、アプリケーションバージョンが要求ヘッダーの以下のヘッダーフィールドで提示される必要があります。

Mobile-Device-Type

Mobile-Application-Name

Mobile-Application-Version

注:
Enterprise Gateway ルールで設定されていないデバイスタイプまたはアプリケーション名が要求に含まれている場合、要求は Enterprise Gateway Server によって許可されます。同様に、無効な形式のバージョンが要求で指定されている場合、要求はサーバによって許可されます。フィルタで指定された条件と一致する場合にのみ、要求はルールに違反します。

モバイルアプリケーションを制御するには

1. Enterprise Gateway Server として機能する Integration Server 上で Integration Server Administrator を開きます。

2. [**セキュリティ**] > [Enterprise Gateway ルール] に移動します。

3. [**モバイルアプリケーション対策オプション**] を選択します。

4. [**デバイスタイプの編集**] をクリックし、制限するモバイルデバイスの名前を 1 行に 1 つずつ入力します。デバイスタイプを入力した後で、[**変更内容の保存**] をクリックします。

5. [**モバイルアプリケーションの編集**] をクリックし、制限するモバイルアプリケーションの名前を 1 行に 1 つずつ入力します。デバイスタイプを入力した後で、[**変更内容の保存**] をクリックします。

6. [**Enterprise Gateway ルールに戻る**] をクリックします。

7. [Enterprise Gateway ルール] ページで、[**ルールの作成**] を選択して新しいルールを作成するか、既存のルールをリストから選択します。

8. ルールの作成ページまたはルールの編集ページで、ページの [**モバイルアプリケーションの保護フィルタ**] 部分にスクロールします。

9. デバイスタイプ、モバイルアプリケーション名、条件、アプリケーションバージョンを選択します。

   バージョンは次の形式で指定します。

   major-version.[*minor-version*.[*sub-minor-version*.[*patch*]]]

   たとえば、XYZ モバイルデバイスから送信される BigApp モバイルアプリケーションの古いバージョン (3.0 よりも前) からの要求を拒否するには、次のように指定します。

フィールド	値
[デバイスタイプ]	XYZ
[モバイルアプリケーション]	BigApp
[条件]	<
[モバイルアプリケーションバージョン]	3.0

   XYZ モバイルデバイスから送信される BigApp モバイルアプリケーションの 3.0 のすべてのバージョン (3、3.0、3.0.0、3.0.0.0) からの要求を許可するには、次のように指定します。

フィールド	値
[デバイスタイプ]	XYZ
[モバイルアプリケーション]	BigApp
[条件]	=

フィールド	値
[モバイルアプリケーションバージョン]	3.0

10. 別の条件を追加するには、[**追加**] をクリックして前の手順を繰り返します。

11. [**変更内容の保存**] をクリックします。

Enterprise Gateway についてよく寄せられる質問

ここでは、Enterprise Gateway についてよく寄せられる質問にお答えします。

Enterprise Gateway 外部ポートで HTTPS を使用するように定義した場合、Enterprise Gateway 登録ポートも HTTPS ポートとして定義する必要がありますか。
必要ありません。外部ポートと登録ポートは独立して動作します。

Enterprise Gateway Server と内部サーバの間に、接続をいくつ登録すればよいのでしょうか。
予測されるトランザクションの負荷とサイズによります。Enterprise Gateway Server と内部サーバとの接続は、要求が内部サーバに書き込まれている間または応答が内部サーバから返されている間以外は使用可能です。言いかえれば、Enterprise Gateway の接続は送受信に左右されるということです。したがって、サイズの大きなトランザクションが同時に発生する可能性がある場合は、それに従って登録接続数を増やしてください。

内部サーバで使用できる登録接続が足りなくなると、次のようなエラーメッセージが表示されます。

```
number 要求が登録接続に対して待機しています。
```

接続ごとに、内部サーバの共通スレッドプールから、または内部リスナーのプライベートスレッドプールが定義されている場合にはこのスレッドプールから、スレッドが 1 つ消費されます。消費されたスレッドは、Enterprise Gateway Server からの要求の処理にしか使用できません。

内部登録リスナー用のプライベートスレッドプールを定義してある場合、[**最大接続数**] ボックスに指定できる接続の数は、このリスナーのプライベートスレッドプールで許可されている最大スレッド数に制限されます。

複数の内部登録リスナーがあり、それぞれに専用のプライベートスレッドプールが設定されている場合、各内部登録リスナーに対して同じルールが適用されます。

内部登録リスナーにプライベートスレッドプールを定義していない場合、[**最大接続数**] ボックスの妥当な最大値は、[設定] > [リソース] ページの [**サーバスレッドプール**] の [最大スレッド数] ボックスで指定したサーバスレッド数の 75% です。複数の内部登録リスナーがあり、それらのいずれにもプライベートスレッドプールが設定されていない場合は、これらのリスナーの [**最大接続数**] ボックスに指定されているすべての接続の合計が、[**サーバスレッドプール**] の [最大スレッド数] で指定されているサーバスレッド数の 75% を超えないようにする必要があります。

スレッドは、ファイアウォール、ネットワークの異常または例外によって閉じられない限り、開いたままになります。

Enterprise Gateway Server には継続性はありますか。

ありません。Enterprise Gateway Server は受信要求にとってのネットワーク上のホップに過ぎません。

外部クライアントの SSL クレデンシャルを行う必要があります。どこに認証を設定すればよいのでしょうか。

次の表は、内部サーバがクライアント認証を実行する、デフォルトの Enterprise Gateway 設定の認証を設定する場所を示しています。クライアント認証を Enterprise Gateway Server でも実行する必要がある場合は、617 ページの「Enterprise Gateway Serverでのクライアント認証の実行 」を参照してください。

Enterprise Gateway Server	内部サーバ
Enterprise Gateway外部ポート	
■ Enterprise Gateway Server のサーバ認証と秘密鍵を含むキーストアを設定します。	
■ Enterprise Gateway Server によって信用される認証局の認証を含むトラストストアを設定します。 Enterprise Gateway Server は、外部クライアントによって送信された認証が、このトラストストアの CA によって署名されていることを確認します。 このトラストストアは、内部サーバ上のトラストストアと同じである必要があります。	■ 内部サーバによって信用される認証局の認証を含むトラストストアを設定します。 内部サーバは、外部クライアントによって (Enterprise Gateway Server から) 送信された認証が、このトラストストアの CA によって署名されていることを確認します。 このトラストストアは、Enterprise Gateway 外部ポートのトラストストアと同じである必要があります。
■ (オプション) 外部ユーザの公開認証をインポートして、Enterprise Gateway Server のユーザにマッピングします。 この操作は、内部サーバに加えて Enterprise Gateway Server でもクライアント認証を実行する場合にのみ行います。 クライアント認証を Enterprise Gateway Server と内部サーバの両方で実行する場合は、認証マッピングが両方のサーバで同じであることを確認します。	■ 外部ユーザの公開認証をインポートして、内部サーバのユーザにマッピングします。 クライアント認証を Enterprise Gateway Server と内部サーバの両方で実行する場合は、認証マッピングが両方のサーバで同じであることを確認します。
Enterprise Gateway登録ポート (HTTPS)	
■ 内部サーバの公開認証をインポートして、Administrator 特権を持つユーザにマッピングします。	■ 内部サーバの認証と秘密鍵を含むキーストアを設定します。
■ 内部サーバの CA 認証が、登録ポートのトラストストアに存在することを確認します。	■ 登録ポートの CA 認証が、内部サーバのトラストストアに存在することを確認します。

Enterprise Gateway Server は送信プロキシサーバとしても使用できますか。

できません。Enterprise Gateway Server を通過する要求は、外部クライアントから内部サーバへの受信要求、およびその要求に対して内部サーバから外部クライアントに返す応答だけです。内部サーバからの要求のうち、応答としての要求でないものは、直接外部クライアントに向けられます。

Enterprise Gateway Server および内部サーバでは、どの認証モードを使用すればよいのでしょうか。
認証モードは、サーバがクライアント要求の認証に使用するメソッドです。デフォルトの Enterprise Gateway 設定では、Enterprise Gateway Server は認証情報を外部クライアントから受け取り、認証を実行する内部サーバに渡します。

内部サーバと Enterprise Gateway 外部ポートには同じ認証モードを指定するようにしてください。たとえば、内部サーバの認証モードが [**必須**] の場合、外部ポートの認証モードも [**必須**] とする必要があります。このように設定することにより、外部クライアントの認証が、Enterprise Gateway Server から内部サーバに必ず渡されるようになります。

一方、Enterprise Gateway 登録ポートの認証モードは、内部サーバまたは Enterprise Gateway 外部ポートの認証モードと一致する必要はありません。

クライアント認証を Enterprise Gateway Server で実行する必要がある場合は、617 ページの「Enterprise Gateway Serverでのクライアント認証の実行 」を参照してください。

Enterprise Gateway は FTP プロトコルをサポートしていますか。
サポートしていません。HTTP および HTTPS のみをサポートしています。

SOCK および SSLSOCK プロトコルはサポートされていますか。
サポートされていません。これらのプロトコルは古いリリースで使用されていた独自仕様のプロトコルです。7.1 リリース以降、SOCK および SSLSOCK の代わりに HTTP および HTTPS が使用されています。

Enterprise Gateway Server 上で要求をフィルタすることは可能ですか。
はい。Enterprise Gateway ルールを使用すると、要求サイズ、要求タイプ、呼び出し対象のリソース名など、いくつかの要素に基づいて要求をフィルタできます。

Enterprise Gateway サーバと内部サーバ間のネットワーク接続障害の後、内部サーバのポートを再有効化する必要があります。どのようにすればこの問題を避けることができますか。
内部サーバで、watt.server.rg.internalregistration.timeout の値をデフォルト値 0 以外の値に変更します。これは、Enterprise Gateway サーバへの接続が無応答でも、内部サーバが接続を閉じないことを示します。さらに、内部サーバの watt.server.rg.internalregistration.timeout の値が Enterprise Gateway サーバの watt.net.socketpool.sweeperInterval の値より大きくなるようにしてください。

どのようにすれば、Enterprise Gateway サーバのクライアント要求が内部サーバへの接続まで無期限に待機することを避けられますか。
Enterprise Gateway サーバがクライアント要求を内部サーバに渡すことができるのは、内部サーバへの接続が使用できる場合だけです。内部サーバへの接続がないか、すべての内部サーバ接続が他の要求の処理でビジー状態である場合、Enterprise Gateway サーバの外部ポートから受信したクライアント要求は待機状態になります。接続が使用可能になるまでに数分経過し、クライアントが要求をタイムアウトにする可能性があります。この待機期間内にクライアントは Enterprise Gateway サーバからフィードバックを受信しないため、クライアントサイドの再試行メカニズムによりクライアント要求が再送され、トランザクションが重複する可能性があります。

Integration Server のサーバ設定プロパティ watt.server.rg.internalsocket.timeout は、Enterprise Gateway サーバが内部サーバへの接続までの待機時間を制御し、要求元クライアントに HTTP 500 内部サーバエラーを返します。指定したタイムアウト期間内に内部サーバへの接続が使用できるようになった場合、Enterprise Gateway サーバは要求を内部サーバに転送します。タイムアウトが経過する前に接続が使用できるようにならない場合、Enterprise Gateway サーバは要求元クライアントに HTTP 500 内部サーバエラーを返します。

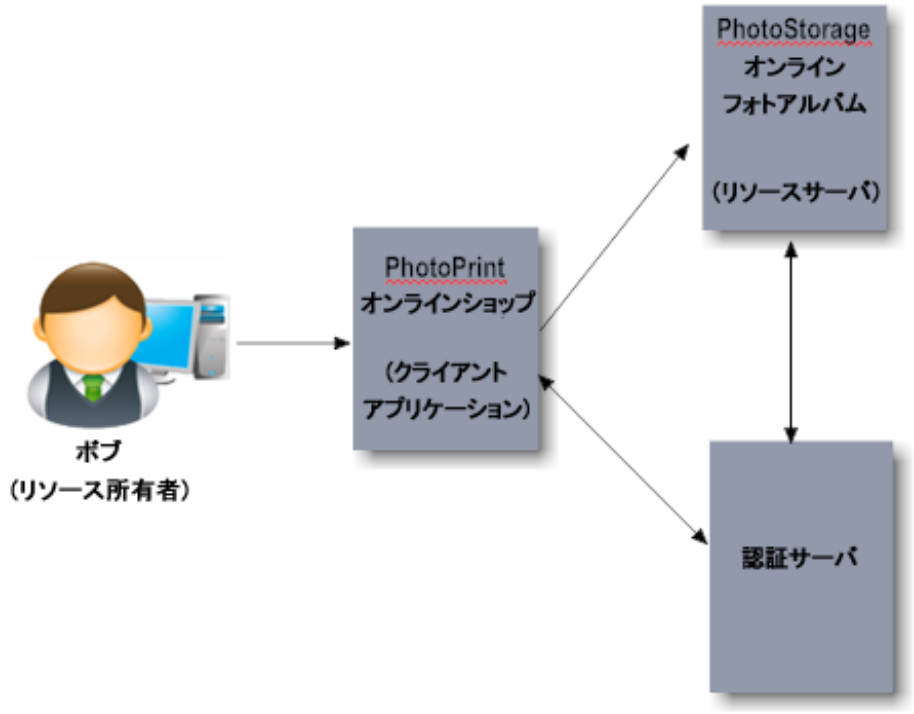
35 OAuth の設定

- OAuth とは 638
- Integration Server での OAuth の使用 639
- OAuth クライアントタイプ 640
- Integration Server でサポートされる承認の譲渡タイプ 641
- Integration Server OAuth サービス 649
- OAuth 機能の使用に関する重要な考慮事項 650
- OAuth のための Integration Server の設定 651
- リソースサーバとしての Integration Server の使用について 670
- 外部認証サーバの使用 670

OAuth とは

OAuth 2.0 Authorization Framework (OAuth) によって、サードパーティのクライアントアプリケーション (クライアント) とのプライベートリソース (データまたはサービス) の共有が容易になります。OAuthセッションでは、プライベートリソースはリソースサーバに格納され、リソースの所有者 (リソース所有者) はリソースにアクセスする権限をクライアントアプリケーションに譲渡します。通常、リソース所有者は人ですが、アプリケーションである場合もあります。リソース所有者が権限を譲渡すると、OAuth 認証サーバによって、アクセストークンがクライアントアプリケーションに発行されます。 クライアントアプリケーションがアクセストークンをリソースサーバに渡すと、リソースサーバは認証サーバと通信してトークンの妥当性検査を行い、有効な場合は、リソースへのアクセスを可能にします。

次の例は、OAuthセッションに関連する役割を示しています。この例では、ボブ氏は、PhotoPrint サービス (クライアントアプリケーション) を使用し、PhotoStorage Web サイト (リソースサーバ) に格納されている写真にアクセスして印刷するリソース所有者です。PhotoPrint は、ボブ氏のデバイス (電話、ラップトップなど) で実行されるアプリケーションを提供します。ボブ氏は、そのアプリケーションを使用してプロセスを開始します。PhotoPrint から PhotoStorage 認証サーバに要求が送信されます。認証サーバは、ボブ氏からの承認を要求し、PhotoPrint にトークンを発行します。これで、PhotoPrint は PhotoStorage にあるボブ氏の写真にアクセスできます。



OAuth の詳細な説明はこのマニュアルの範囲外ですが、別途入手できます。OAuth プロトコルの詳細については、OAuth 2.0 Authorization Framework を参照してください。

Integration Server での OAuth の使用

Integration Server は、OAuth クライアント、認証サーバまたはリソースサーバとして使用できます。Integration Server Administrator によって、これらの役割を作成および管理するために開発者が使用できる OAuth 設定機能が提供されます。OAuth 機能を有効にするには、Integration Server に Software AG からライセンスを取得している必要があります。ライセンスの詳細については、650 ページの「OAuth 機能の使用に関する重要な考慮事項」を参照してください。

OAuth クライアントとしての Integration Server

Integration Server が OAuth クライアントアプリケーションである場合は、pub.client.oauth:executeRequest サービスを使用して、Facebook、Google、Twitter、Integration Server などのプロバイダからのリソースにアクセスします。pub.client.outh:executeRequest サービスを使用したリソースへのアクセスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

認証サーバとしての Integration Server

Integration Server が認証サーバとして機能する場合、クライアントアプリケーションから承認要求を受信します。クライアントアプリケーションでは、pub.oauth:authorize サービスを呼び出すことによって要求を開始します。要求の承認のために、認証サーバによってクライアントアプリケーション、リソースサーバ、リソース所有者間のインタラクションが処理されます。認証サーバとしての Integration Server の設定の詳細については、651 ページの「OAuth のための Integration Server の設定」を参照してください。

Integration Server が認証サーバとして機能する場合、アクセストークンはベアラートークンとして発行されます。ベアラートークンは、アクセストークンを所有している任意のパーティ (ベアラー) にトークンの使用を許可するアクセストークンです。認証サーバには、発行するベアラートークンに関する情報 (ユーザ情報を含む) が保持されます。クライアントによってベアラートークンがリソースサーバに提示されると、トークンはリソースサーバから認証サーバに送信され、トークンが有効であり、要求されたサービスがアクセストークン発行のスコープ内であることが確認されます。スコープとは、クライアントがリソース所有者に代わってアクセスできるフォルダおよびサービス (リソース) の定義のことです。

フォルダおよびサービスにアクセスする権限がユーザにある場合、リソースサーバによって要求が処理されます。リソースにアクセスする特権がユーザにない場合、リソースサーバによって要求は拒否されます。ユーザ特権の詳細については、101 ページの「ユーザとグループの管理」を参照してください。

外部認証サーバとしての Integration Server

Integration Server の認証サーバによって生成されたアクセストークンを使用して、他のベンダーのサーバのリソースにアクセスできます。つまり、Integration Server 認証サーバは Integration Server ではないリソースサーバからのアクセストークンのイントロスペクション要求に応答できます。これは RFC 7662 (OAuth 2.0 Token Introspection) の Integration Server サポートの一部です。

pub.oauth* サービスは OAuth 2.0 認証サーバとして使用される Integration Server とやりとりするためのサービスをクライアントに提供します。このサービスでは、トークンの要求、トークンがまだアクティブであるかどうかの判断、アクセストークンのリフレッシュ、アクセストークンの取り消しなどが行われます。Integration Server 認証サーバのイントロスペクションエンドポイントは次の URL です。

https://*host*:*port*/invoke/pub.oauth/introspectToken

host:*port* にある Integration Server 認証サーバの pub.oauth:introspectToken が呼び出されます。

OAuth 用の組み込みサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

リソースサーバとしての Integration Server

Integration Server がリソースサーバとして機能する場合、アクセストークンを含むクライアントアプリケーションからの要求を受信します。リソースサーバから認証サーバに、アクセストークンおよびユーザの妥当性検査が要求されます。トークンが有効であり、フォルダおよびサービスにアクセスする特権がユーザにある場合、リソースサーバによって要求が処理されます。リソースサーバと認証サーバは、同じ Integration Server インスタンスであるか、異なる Integration Server インスタンスです。認証サーバはサードパーティの認証サーバにすることもできます。リソースサーバとしての Integration Server の使用の詳細については、670 ページの「リソースサーバとしての Integration Server の使用について」を参照してください。サードパーティ認証サーバの設定の詳細については、670 ページの「外部認証サーバの使用」を参照してください。

OAuth クライアントタイプ

OAuth 2.0 Authorization Framework は 2 つのタイプのクライアントを定義します。

- 機密クライアントは、アクセストークンを取得するために認証サーバにクライアント ID とクライアントシークレットを提示するクライアントです。機密クライアントは、認証サーバ上のアカウントに対応しています。ユーザアカウントに対する適切なクレデンシャル (クライアント ID とシークレット) がクライアントにない場合、認証サーバからクライアントにアクセストークンは譲渡されません。

  Integration Server Administrator で機密クライアントを指定するには、[**タイプ**] を [**機密**] に設定します。詳細については、655 ページの「クライアントの登録」を参照してください。

- 公開クライアントは、識別にクライアント ID のみを使用し、他のクレデンシャルは使用しないクライアントです。公開クライアントは、通常は JavaScript などのスクリプト言語を使用してブラウザに実装されます。認証サーバによって他のクレデンシャルが要求されないため、有効なクライアント ID を持つ任意のクライアントにアクセストークンが譲渡されます。

  Integration Server Administrator で公開 (暗黙的) クライアントを指定するには、[**タイプ**] を [**公開**] に設定します。詳細については、655 ページの「クライアントの登録」を参照してください。

Integration Server でサポートされる承認の譲渡タイプ

リソース所有者、クライアントアプリケーション、認証サーバおよびリソースサーバ間の承認要求と応答のフローは、OAuth セッションによって定義される承認の譲渡タイプによって異なります。Integration Server では、以下の承認の譲渡タイプがサポートされています。

- 認証コードの譲渡
- 暗黙的譲渡
- リソース所有者のパスワードクレデンシャルの譲渡
- クライアントクレデンシャルの譲渡

Integration Server に登録された各 OAuth クライアントに対して、クライアントが使用できる OAuth 譲渡タイプを指定する必要があります。

認証コードの譲渡

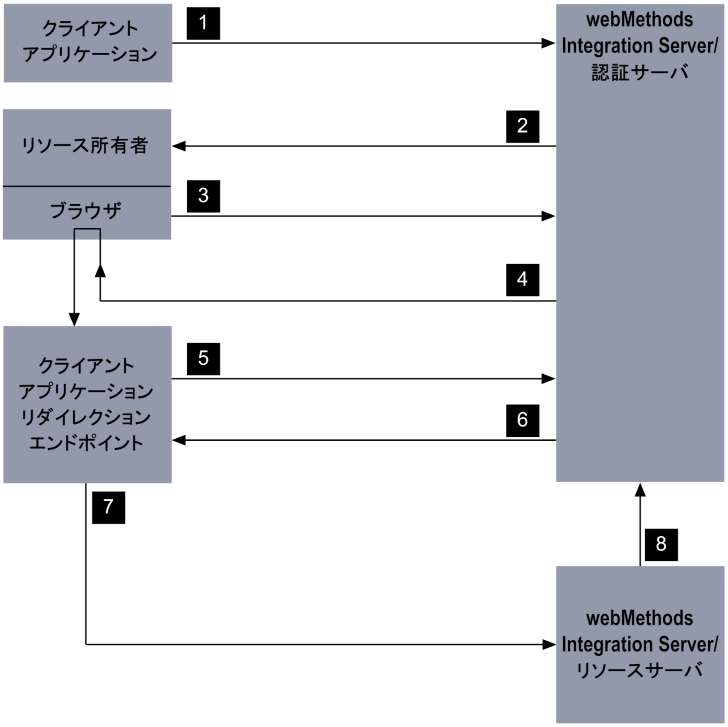
認証コードの譲渡タイプは、認証サーバ上にクレデンシャルを持つクライアントを認証してアクセスを提供するために使用されます。この譲渡タイプには、アクセストークンを取得する前に、認証サーバに対して認証するクライアントが必要です。

認証コードの譲渡タイプを使用して、機密クライアントおよび公開クライアントを認証し、アクセスを提供します。

認証コードの譲渡タイプを使用する場合、認証サーバは、アクセストークンと共にリフレッシュトークンをクライアントアプリケーションに発行できます。 リフレッシュトークンによって、クライアントはリソース所有者からの追加承認を要求しなくても、新しいアクセストークンを取得できます。アクセストークンが期限切れになったら、クライアントアプリケーションは pub.oauth:getToken サービスを使用してリフレッシュトークンを認証サーバに渡し、新しいアクセストークンを要求できます。

次の図は、Integration Server 認証サーバが認証コードの譲渡プロセスに参加する仕組みを示しています。

認証コードの譲渡フロー



次の表では、認証コードの譲渡フローの各ステップについて説明します。

ステージ	説明
1	クライアントアプリケーションは、pub.oauth:authorize サービスを呼び出してリソース所有者のデータへのアクセスを要求することで、プロセスを開始します。 公開クライアントに PKCE が必要な場合、クライアントは `code_verifier` 値を作成する必要があります。クライアントは pub.oauth:authorize サービスに次の追加値を渡す必要があります。 ■ *code_challenge*。`code_verifier` を変換した値です。 ■ *code_challenge_method*。`code_verifier` の変換メソッドです。 PKCE の詳細については、643 ページの「認証コードの譲渡で PKCE を使用する」を参照してください。
2	pub.oauth:authorize サービスによって要求の妥当性検査が行われます。有効な場合、サービスでは、指定されたスコープ内でクライアントアプリケーションがアクセスを要求していることをリソース所有者に通知する HTML ページで応答します。リソース所有者は、HTML ページを使用して要求を承認または拒否します。 Integration Server は指定された *code_challenge* および *code_challenge_method* の値を OAuthPKCE キャッシュに保存します。

ステージ	説明
3	リソース所有者が要求を承認すると、承認ページによって Integration Server で内部サービスが呼び出されます。リソース所有者が要求を拒否した場合は、エラーが返されます。
4	Integration Server によって、クライアントアプリケーションに対して認証コードが生成されます。サーバは、HTTP リダイレクトを使用して、クライアントアプリケーションによって指定されたリダイレクト URI に認証コードを送信するようにリソース所有者のブラウザに指示します。
5	クライアントアプリケーションのリダイレクト URI にあるサービスによって、認証コードは Integration Server 上の pub.oauth:getToken サービスに渡され、アクセストークンの認証コードが交換されます。また、Integration Server はクライアントが認証コードの譲渡が使用できるようにします。 PKCE が使用されている場合、クライアントは *code_verifier* 値も pub.oauth:getToken に指定する必要があります。
6	Integration Server によって、アクセストークンがクライアントアプリケーションに発行されます。リフレッシュトークンをクライアントに発行するように設定されている場合は、認証サーバはリフレッシュトークンも発行します。 PKCE が使用されている場合、Integration Server は *code_challenge_method* を指定された *code_verifier* に適用します。ステージ 1 で指定された *code_challenge* に一致する場合、Integration Server はアクセストークンを発行します。
7	クライアントアプリケーションは、アクセストークンを使用してリソースサーバのリソースを要求します。
8	リソースサーバは認証サーバをチェックして、アクセストークンが有効であることを確認します。アクセストークンが有効である場合、リソースサーバは、要求されたサービスがアクセストークンが発行されたスコープ内にあることを確認します。また、リソースの所有者が要求されたリソースへのアクセス権限を持つことも確認します。

認証コードの譲渡で PKCE を使用する

PKCE (Proof Key for Code Exchange) は、認証コード横取り攻撃を回避する OAuth 標準です。認証コードの譲渡を使用する公開 OAuth クライアントはこの攻撃に対して脆弱です。PKCE を使用して防御してください。

PKCE 標準は https://datatracker.ietf.org/doc/html/rfc7636 で定義されています。PKCE 標準には、クライアントがどのようにこれらのエンドポイントに渡される追加入力を生成するかが記述されています。

グローバル設定 [**PKCE を必須にする**] が [はい] に設定され、クライアントが認証コードの譲渡タイプを使用する場合、PKCE が公開クライアントに必要です。

PKCE が公開 OAuth クライアントに必要であり、Integration Server が認証サーバである場合、公開 OAuth クライアントは `code_verifier` 値と呼ばれるシークレットを作成し、変換する必要

があります。クライアントは変換されたバージョン (*code_challenge*)、およびオプションとして、変換メソッド (*code_challenge_method*) を pub.oauth:authorize サービスに提供する必要があります。

code_challenge_method の値は `S256` および `plain` です。*code_challenge_method* が `S256` である場合、*code_challenge* は `code_verifier` 値の Base64 でエンコードされた SHA-256 ハッシュです。デフォルト値は `plain` です。*code_challenge_method* が `plain` である場合、*code_challenge* 値は `code_verifier` と同一です。

注:
com.softwareag.util.PKCE.createCodeVerifier() メソッドを使用して、`code_verifier` 値を Base64 でエンコードされた SHA-256 ハッシュに変換して *code_challenge* の値として使用できます。

PKCE が使用されている場合、クライアントが Integration Server でトークンエンドポイントサービス pub.oauth:getToken を呼び出してアクセストークンの認証コードを交換するときに、クライアントは追加入力パラメータ *code_verifier* を指定する必要があります。Integration Server は *code_challenge_method* を適用して、提供された *code_verifier* 値を変換します。変換された値が、認証エンドポイントに送信された *code_challenge* に一致する場合、トークンエンドポイントはアクセストークンを返します。

注:
code_challenge が認証エンドポイント (pub.oauth:authorize) に送信されており、*grant_type* が `authorization_code` である場合、*code_verifier* 入力パラメータが必要です。この要件は、*code_challenge* 値を pub.oauth:authorize に提供した公開クライアントに適用されます。公開クライアントが認証エンドポイントに *code_challenge* を提供すると、認証コードの譲渡プロセスの残りの部分に対して PKCE 処理がトリガーされます。認証サーバとして機能する Integration Server に対して [**PKCE を必須にする**] が選択されているかどうかは関係ありません。

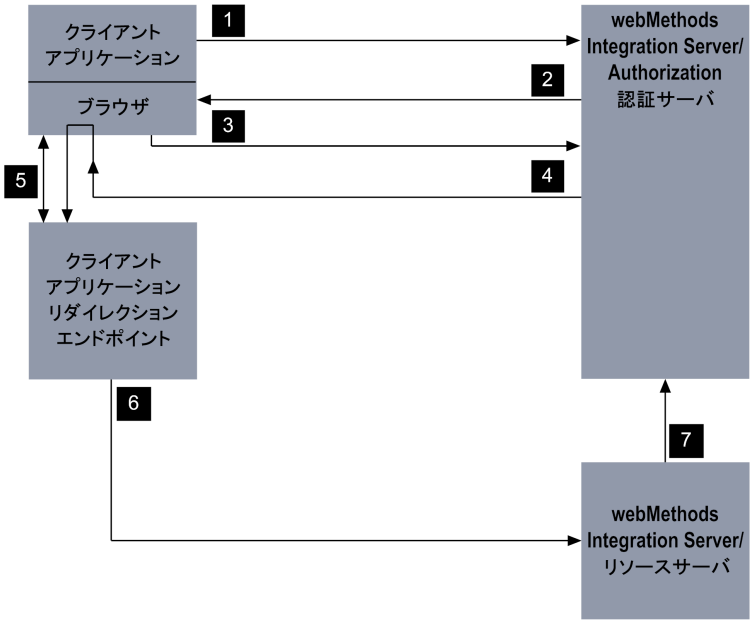
Integration Server は PKCE でクライアントを補助するための Java クラスを備えています。com.softwareag.util.PKCE クラスには、`code_verifier` を作成する createCodeVerifier メソッド、および `code_verifier` 値を、*code_challenge* の値として使用する Base64 でエンコードされた SHA-256 ハッシュに変換する createCodeChallenge メソッドが含まれています。また、このクラスには、*code_verifier* 値が *code_challenge* 値に一致するかどうかを判別する pkceMatch メソッドも含まれています。詳細については、『*webMethods Integration Server Java API Reference*』を参照してください。

暗黙的譲渡

暗黙的譲渡タイプは、ブラウザベースのアプリケーションおよびモバイルアプリケーションを認証するために使用されます。この譲渡タイプでは、認証サーバでの認証がクライアントに要求されないため、認証コードの譲渡よりも安全性が低くなります。また、暗黙的譲渡タイプでは、アクセストークンはリソース所有者のブラウザによって渡されるため、リソース所有者のデバイスでの悪意のあるアプリケーションによる盗難の危険があります。次の図は、Integration Server 認証サーバが暗黙的譲渡プロセスに参加する仕組みを示しています。

暗黙的譲渡タイプを使用して、公開クライアントを認証してアクセスを提供します。

暗黙的譲渡フロー



次の表では、暗黙的譲渡フローの各ステップについて説明します。

ステージ	説明
1	クライアントアプリケーションは、pub.oauth:authorize サービスを呼び出してリソース所有者のデータへのアクセスを要求することで、プロセスを開始します。
2	pub.oauth:authorize サービスによって要求の妥当性検査が行われます。有効な場合、サービスでは、指定されたスコープ内でクライアントアプリケーションがアクセスを要求していることを所有者に通知する HTML ページで応答します。リソース所有者は、HTML ページ上のフォームを使用して要求を承認または拒否します。
3	リソース所有者が要求を承認すると、承認ページによって Integration Server で内部サービスが呼び出されます。リソース所有者が要求を拒否した場合は、エラーが返されます。
4	Integration Server によって、リソース所有者のブラウザがクライアントアプリケーションのリダイレクトエンドポイントにリダイレクトされ、アクセストークンがリダイレクト URI 上のフラグメントとして付加されます。
5	クライアントアプリケーションがアクセストークンを抽出するまで、リソース所有者のブラウザでアクセストークンはメモリ内に保持されます。
6	クライアントアプリケーションは、アクセストークンを使用してリソースサーバのリソースを要求します。
7	リソースサーバは認証サーバをチェックして、アクセストークンが有効であることを確認します。アクセストークンが有効である場合、リソースサーバは、要求されたサービスがアクセストークンが発行されたスコープ内であることを確認します。ま

ステージ	説明
	た、リソースの所有者が要求されたリソースへのアクセス権限を持つことも確認します。

リソース所有者パスワードのクレデンシャルの譲渡

リソース所有者パスワードのクレデンシャル (ROPC) の譲渡により、リソース所有者は信頼できるクライアントに、アクセストークンを取得するために使用できるクレデンシャルを提供することができます。クライアントは、アクセストークンを使用してリソースサーバのリソースにアクセスできます。ROPC の譲渡は、リソース所有者とクライアントに信頼関係が存在する場合、または他の承認譲渡タイプが使用できない場合に、リソースにアクセスを提供することを目的としています。

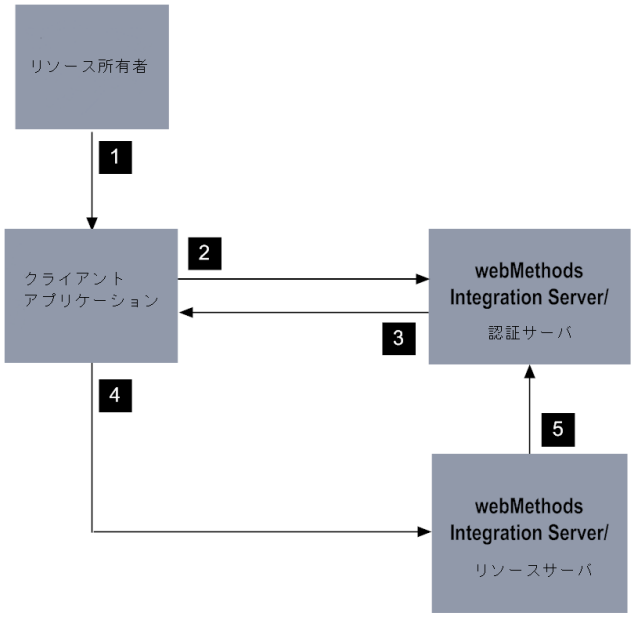
ROPC 譲渡タイプがあれば、クライアントはリソース所有者の承認や、リダイレクトエンドポイントを必要としません。代わりに、クライアントはトークンエンドポイントを認証サーバに直接呼び出し、認証のためにリソース所有者のクレデンシャルを提示します。トークンエンドポイントは、クレデンシャルを認証し、アクセストークンを発行します。

ROPC 譲渡タイプを使用する場合、認証サーバは、アクセストークンと共にリフレッシュトークンをクライアントアプリケーションに発行できます。 リフレッシュトークンによって、クライアントはリソース所有者からの追加承認を要求しなくても、新しいアクセストークンを取得できます。アクセストークンが期限切れになったら、クライアントアプリケーションは pub.oauth:getToken サービスを使用してリフレッシュトークンを認証サーバに渡し、新しいアクセストークンを要求できます。

ROPC の譲渡タイプを使用して、機密クライアントおよび公開クライアントを認証し、アクセスを提供します。

ROPC 譲渡タイプは Integration Server がサポートするすべての譲渡タイプの中で安全性が最も低いです。

リソース所有者のパスワードクレデンシャルの譲渡



次の表では、ROPC 譲渡フローの各ステップについて説明します。

ステージ	説明
1	リソース所有者はクライアントアプリケーションにクレデンシャルを提供します。
2	クライアントアプリケーションは、リソース所有者のクレデンシャルを認証サーバのトークンエンドポイント pub.oauth:getToken サービスに提供します。
3	認証サーバで、pub.oauth:getToken サービスは提供されたクレデンシャルを認証し、クレデンシャルがリソース所有者のパスワードクレデンシャル (ROPC) の譲渡の使用を許可された機密クライアントに属することを検証します。また、サービスはクライアント要求で指定されたスコープがクライアントに対して許可されていることも確認します。認証サーバはクライアントにアクセストークンを返します。リフレッシュトークンをクライアントに発行するように設定されている場合は、認証サーバはリフレッシュトークンも返します。
4	クライアントアプリケーションは、アクセストークンを使用してリソースサーバのリソースを要求します。
5	リソースサーバは認証サーバをチェックして、アクセストークンが有効であることを確認します。アクセストークンが有効である場合、リソースサーバは、要求されたサービスがアクセストークンが発行されたスコープ内であることを確認します。また、リソースの所有者が要求されたリソースへのアクセス権限を持つことも確認します。

クライアントクレデンシャルの譲渡

クライアントクレデンシャルの譲渡は、認証サーバ上にクレデンシャルを持つクライアントを認証してアクセスを提供するために使用されます。クライアントクレデンシャルの譲渡により、ク

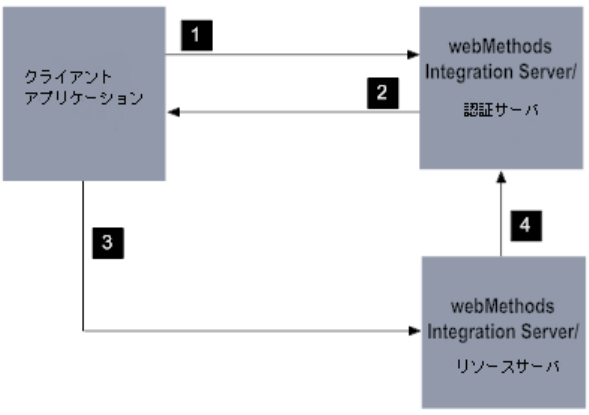
ライアントはクレデンシャルを使用して、リソースにアクセスするために使用するアクセストークンを取得することができます。クライアントクレデンシャルの譲渡は、既にクライアント制御にあるリソースにアクセスする場合、リソース所有者がクライアントがリソースにアクセスすることを信用している場合、またはマシン間のやり取りに対して頻繁に使用されます。マシン間のやり取りの例としては、リソース所有者のデバイス上で実行するバックグラウンドプロセス、および OAuth で保護されたリソースにアクセスする必要のあるバックグラウンドプロセスがあります。

認証コードの譲渡タイプとは異なり、クライアントクレデンシャルの譲渡を使用しているクライアントは、アクセストークンを取得する前にリソース所有者の承認や、リダイレクトエンドポイントを必要としません。代わりに、クライアントはクレデンシャルをエンドポイントに直接提示し、アクセストークンを要求します。トークンエンドポイントは、クレデンシャルを認証し、アクセストークンを発行します。リフレッシュトークンがクライアントクレデンシャルの譲渡によってサポートされていないので、トークンエンドポイントはリフレッシュトークンを発行しません。

クライアントクレデンシャルの譲渡タイプは、認証コードの譲渡タイプよりも安全性が低いです。

クライアントクレデンシャルの譲渡を使用するクライアントは、認証サーバ上にクレデンシャルが必要であり、このことはクライアントが機密クライアントである必要があることを意味します。クライアントクレデンシャルの譲渡を使用するクライアントを作成する場合、クライアントに必要なリソースにアクセスする権限を与える必要があります。このことは、Integration Server がリソース所有者を知っているほかの譲渡タイプと異なります。なぜならリソース所有者は要求を承認するときにクレデンシャルを提供している (認証コードの譲渡および暗黙的譲渡) か、クライアントがリソース所有者のクレデンシャル (ROPC 譲渡) を提供しているからです。クライアントクレデンシャルの譲渡では、クライアントはリソース所有者の承認を必要としないので、Integration Server は所有者を知りません。このため、クライアントがリソースにアクセスする権限を持っている必要があります。必要なアクセス権限を持つ 1 つ以上のユーザグループに、クライアントを追加する必要があります。

クライアントクレデンシャルの譲渡フロー



次の表では、クライアントのクレデンシャルの譲渡フローの各ステップについて説明します。

ステージ	説明
1	クライアントアプリケーションは、認証サーバでトークンエンドポイントである pub.oauth:getToken サービスにクレデンシャルを提示して、プロセスを開始します。
2	認証サーバで pub.oauth:getToken サービスはクライアントを認証し、クレデンシャルがクライアントクレデンシャルの譲渡の使用を許可された機密クライアントに属することを検証します。また、サービスはクライアント要求で指定されたスコープがクライアントに対して許可されていることも確認します。認証サーバはクライアントにアクセストークンを返します。 **注:** リフレッシュトークンはクライアントクレデンシャルの譲渡に対して返されていません。
3	クライアントアプリケーションは、アクセストークンを使用してリソースサーバのリソースを要求します。
4	リソースサーバは認証サーバをチェックして、アクセストークンが有効であることを確認します。アクセストークンが有効である場合、リソースサーバは、要求リソースサービスがアクセストークン発行のスコープ内であることを確認します。

Integration Server OAuth サービス

Integration Server は WmPublic パッケージの OAuth に使用するパブリックサービスを提供します。

次の表は、リソースサーバへのクライアントアプリケーションのアクセスを認可するために使用するサービスを示しています。

サービス	説明
pub.oauth:authorize	クライアントアプリケーションから Integration Server 認証サーバへの承認要求を開始します。
pub.oauth:getAccessToken	Integration Server 認証サーバからのアクセストークンを要求します。要求には、認証サーバによってリダイレクト URI に送信される認証コードが含まれています。 認証サーバによって要求の妥当性検査が行われ、アクセストークンおよびリフレッシュトークン (オプション) が生成されます。トークンは、クライアント識別子、期限切れまでの時間、スコープと共に認証サーバのキャッシュに格納されます。 **注:**

サービス	説明
	pub.oauth:getAccessToken サービスは廃止されました。代わりに pub.oauth:getToken を使用してください。
pub.oauth:getToken	Integration Server 認証サーバからのアクセストークンを要求します。要求には、認証サーバによってリダイレクト URI に送信される認証コードが含まれています。 認証サーバによって要求の妥当性検査が行われ、アクセストークンおよびリフレッシュトークン (オプション) が生成されます。トークンは、クライアント識別子、期限切れまでの時間、スコープと共に認証サーバのキャッシュに格納されます。 **注:** pub.oauth:getToken サービスによって、廃止された pub.oauth:getAccessToken サービスおよび pub.oauth:refreshAccessToken サービスが置き換えられます。
pub.ouath:introspectToken	認証サーバとして使用される Integration Server によって生成されたアクセストークンまたはリフレッシュトークンがアクティブであるかどうかをチェックします。
pub.oauth:refreshAccessToken	Integration Server 認証サーバに要求を送信して、アクセストークンをリフレッシュします。 **注:** pub.oauth:refreshAccessToken サービスは廃止されました。代わりに pub.oauth:getToken を使用してください。
pub.oauth:removeExpiredAccessTokens	データベースから期限切れのアクセストークンを削除します。
pub.oauth:revokeToken	Integration Server 認証サーバでトークンを取り消します。

Integration Server OAuth サービスの使用の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

OAuth 機能の使用に関する重要な考慮事項

OAuth 機能 を使用する場合は、以下の点に留意してください。

- webMethods Enterprise Gateway を使用して外部クライアントからの要求を処理している場合、Integration Server Administrator での OAuth 設定 ([セキュリティ] > [**OAuth**]) は Enterprise Gateway Server として機能していない Integration Server でのみ使用できます。ファイアウォールの内側にある内部サーバによって、すべての OAuth 要求が処理されます。webMethods Enterprise Gateway の詳細については、593 ページの「webMethods Enterprise Gateway の設定 」を参照してください。
- Integration Server で OAuth アクティビティをログに記録するには、セキュリティロガーを有効にして、認証と承認のセキュリティ領域をログに記録するように設定する必要があります。セキュリティロガーを有効にして、監査するセキュリティ領域を選択する手順については、『*webMethods Audit Logging Guide*』を参照してください。
- エラーログに OAuth 認証エラーを表示するには、watt.server.oauth.log.authErrors サーバ設定パラメータが「true」に設定されている必要があります。デフォルトで、このパラメータは「false」に設定されています。つまり、Integration Server では OAuth 認証エラーがログに書き込まれません。

OAuth のための Integration Server の設定

Integration Server 環境で OAuth の使用を開始する前に、OAuth ソリューションで Integration Server が果たす役割に従って、認証サーバまたはリソースサーバのどちらか一方または両方の設定を指定する必要があります。認証サーバおよびリソースサーバを設定すると、クライアントの登録および OAuth スコープの管理を開始できます。

次の表では、OAuth に Integration Server を設定するための基本ステージについて説明します

ステージ 1 OAuth 設定を設定します。

このステージでは、Integration Server で OAuth 設定を設定します。Integration Server は、特定の OAuth 設定を使用するようにデフォルトで設定されています。これらの設定を、使用するシステム用の設定を反映するように設定する方法の詳細については、652 ページの「OAuth 設定の設定」を参照してください。

注:
このステージは、認証サーバとして使用される Integration Server に主に適用されます。ただし、Integration Server がリソースサーバとして機能している場合、[セキュリティ] > [OAuth] > [OAuth グローバル設定の編集] ページの [**認証サーバ**] フィールドを使用して、認証サーバをリソースサーバに指定する必要があります。

ステージ 2 クライアントを定義します。

このステージでは、認証サーバにアクセスする権限があるクライアントを定義し、どの譲渡タイプが使用できるかを指定します。クライアントの登録、変更、削除の詳細については、655 ページの「クライアントの定義」を参照してください。

認証サーバとして Integration Server を使用する場合、Integration Server とリソースサーバは同じリソース所有者を持ってる必要があります。この要件は、外部認証サーバを使用しているとき、または、すべてのクライアントがクライアントクレデンシャルの譲渡タイプを使用する場合には適用されません。

認証サーバとリソースサーバに Integration Server を使用している場合、一方の Integration Server で client_id 値を定義し、他方の Integration Server に値を展開することができます。

クライアントクレデンシャルの譲渡を使用するクライアントは、クライアントに関連するユーザアカウントが、認証サーバおよびリソースサーバ上にある必要があります。

ステージ 3 スコープを定義します。

このステージでは、アクセスするためにクライアントが使用できるスコープを定義します。スコープの追加、変更、削除の詳細については、662 ページの「スコープの定義」を参照してください。

認証サーバとリソースサーバのスコープ名は同じにする必要があります。スコープ名は各サーバで定義できます。または、認証サーバとリソースサーバに Integration Server を使用している場合、一方の Integration Server でスコープを定義し、他方の Integration Server に値を展開することができます。

ステージ 4 スコープをクライアントに関連付けます (またはその逆)。

このステージでは、スコープをクライアントに関連付けます。スコープとクライアントを関連付けると、各クライアントがアクセスできるスコープが認可されます。スコープとクライアント間の関連付けの追加、削除、表示の詳細については、665 ページの「スコープとクライアントの関連付け」を参照してください。

注:
このステージは、認証サーバとして使用される Integration Server にのみ適用されます。リソースサーバとして使用される Integration Server に対しては、このステージを完了する必要がありません。

ステージ 5 認証サーバによってアクセストークンが譲渡された後で、特定のクライアントアプリケーションがリソースにアクセスできないようにする場合は、以下のいずれかを実行できます。

- 認証サーバで、アクティブなアクセスを削除し、そのクライアントアプリケーションに譲渡されているトークンをリフレッシュします。トークンの表示および削除の詳細については、668 ページの「トークンの管理」を参照してください。
- リソースサーバで、クライアントアプリケーションを無効にします。クライアントアプリケーションを無効にする方法の詳細については、660 ページの「クライアントの有効化および無効化」を参照してください。

OAuth 設定の設定

認証サーバの OAuth グローバル設定によって、OAuth 通信に HTTPS が必要かどうかが制御されます。また、認証コードおよびアクセストークン有効期間のグローバル値も指定できます。有効期間は、グローバルに設定するか、個々のクライアントごとに設定できます。

OAuth 設定を設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**OAuth グローバル設定の編集**] をクリックします。
4. [認証サーバ設定} で、Integration Server が認証サーバとして機能する場合、以下のフィールドを入力します。

フィールド	説明
[HTTP が必要]	要求を認可するために認証サーバが HTTPS 接続を要求するかどうかを示します。 有効 (デフォルト) の場合、認証サーバが HTTPS を使用して pub.oauth サービスを呼び出すことが Integration Server によって要求されます。無効の場合、クライアントアプリケーションが HTTP を使用して pub.oauth サービスにアクセスすることが Integration Server によって許可されます。 [**HTTP が必要**] が有効で、クライアントアプリケーションが HTTP で pub.oauth サービスのいずれかにアクセスすると、Integration Server によって HTTP 400 エラー応答がクライアントに発行され、サービス例外がエラーログに書き込まれます。 **重要:** 開発を簡素化するために、[**HTTP が必要**] を無効にできますが、実稼動では、OAuth Framework に従って HTTPS を使用する必要があります。HTTPS を要求しない場合、アクセストークンは認証サーバによってクリアテキストで送信され、盗難に対して脆弱になります。
[PKCE を必須にする]	認証コードの譲渡を使用するすべての公開 OAuth クライアントに対して PKCE (Proof Key for Code Exchange) が必要かどうかを示します。 すべての公開 OAuth クライアントが `code_challenge` および `code_challenge_method` を認証エンドポイント (pub.oauth:authorize) サービスに提供し、`code_verifier` をトークンエンドポイント (pub.oauth:getToken サービス) に提供する必要がある場合、[**PKCE を必須にする**] チェックボックスをオンにします。 公開 OAuth クライアントが追加入力を提供する必要がないことを示すには、[**PKCE を必須にする**] チェックボックスをオフにします。ただし、`code_challenge` および `code_challenge_method`

フィールド	説明
	を認証エンドポイントに送信するすべてのクライアントは、PKCE 機能を使用して認証コード横取り攻撃を回避します。クライアントからトークンエンドポイントへの後続の要求には、`code_verifier` 入力パラメータを含める必要があります。
[認証コード有効期間]	認証サーバによって発行された認証コードが有効な時間の長さ (秒単位) を指定します。 有効な値は 1～2147483647 です。デフォルト値は 600 です。
[アクセストークン有効期間]	認証サーバによって発行されたアクセストークンが有効な時間の長さ (秒単位) を指定します。

選択項目	目的
[無期限]	アクセストークンが期限切れにならないことを示します。
[**有効期限**] および秒数を入力します。最大値は 2147483647 です。デフォルトは 3600 です。	アクセストークンが有効な時間の長さを指定します。

トークンエンドポイント認証	トークンエンドポイントが認証に対して既存のセッションまたは要求を受け入れるかどうかを指定します。pub.oauth:getToken サービスはトークンエンドポイントとして機能します。クライアントは Integration Server 認証サーバからこのサービスを呼び出し、アクセストークンを要求します。

選択項目	目的
既存のセッションの受け入れ	トークンエンドポイントサービスが、Integration Server にアクティブなセッションがあるクライアントからの要求を受け入れることを示します。これらのクライアントが Cookie 要求ヘッダーに有効なセッションを提示する場合、クライアントは pub.oauth:getToken を使用するためにクレデンシャルを提供する必要はありません。これはデフォルト動作であり、Integration Server の 10.3 のより前のバージョンに存在した動作に一致します。
クレデンシャルの要求	新しいアクセストークンが要求されるたび、または既存のアクセストークンがリフレッシュされるたびに pub.oauth:getToken サービスを呼び出して、クライアントに対し認証要求

フィールド	説明
	ヘッダーのクレデンシャルの提供を要求します。
	注: pub.oauth:getToken サービスによって、廃止された pub.oauth:getAccessToken サービスおよび pub.oauth:refreshAccessToken サービスが置き換えられます。
	注: トークンエンドポイント認証は、認証コードの譲渡タイプ、リソース所有者パスワードクレデンシャルの譲渡タイプ、およびクライアントクレデンシャルの譲渡タイプを使用してクライアントに影響を与えます。

5. [リソースサーバ設定]で、[**認証サーバ**] リストの Integration Server をリーソースサーバとして設定する場合、認証サーバとなるサーバを選択します。

   認証サーバとして Integration Server を使用できます。または、外部認証サーバを使用できます。[**認証サーバ**] リストには、設定されたリモートサーバエイリアスおよび使用可能な外部認証サーバエイリアスが表示されます。

   認証サーバとしてリモート Integration Server を使用する予定であり、認証サーバのエイリアスをまだ定義していない場合、[**認証サーバ**] リンクをクリックして、[**外部サーバ**] > [リモートサーバ] ページに移動します。リモートサーバエイリアスの作成の詳細については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。

   外部認証サーバを使用する予定で、認証サーバのエイリアスをまだ定義していない場合、[**外部認証サーバの追加**] リンクをクリックして [外部認証サーバ] > [追加] ページに移動します。外部認証サーバのエイリアスの作成の詳細については、670 ページの「外部認証サーバの使用」を参照してください。

   Integration Server を認証サーバとしてのみ設定する場合、[**認証サーバ**] フィールドの値は Integration Server によって無視されます。

6. [**変更内容の保存**] をクリックします。

クライアントの定義

保護されたリソースへのアクセスをクライアントアプリケーション (クライアント) が要求できるようにするには、Integration Server Administrator を使用してクライアントを認証サーバに登録する必要があります。

クライアントの登録

クライアントが OAuth で保護されたリソースにアクセスするには、認証サーバとして機能する Integration Server に登録されている必要があります。クライアントを登録する際は、クライアントを機密クライアントまたは公開クライアントに識別し、クライアントが使用できる譲渡タイプを選択し、トークンの有効期限とリフレッシュ情報を指定します。

Integration Server に登録された各 OAuth クライアントに対して、クライアントがサーバにアクセスするのに使用できる OAuth 譲渡タイプを指定する必要があります。クライアントが使用できる譲渡タイプを選択する場合は、以下の情報に留意してください。

- 機密クライアントだけが、クライアントクレデンシャルの譲渡およびリソース所有者のパスワードクレデンシャルの譲渡を使用できます。公開クライアントに対しては、これらの譲渡タイプを選択できません。クライアントが許可されていない譲渡を使用してアクセストークンを要求した場合、Integration Server は要求を拒否し、unauthorized_client OAuth エラーを返します。
- 新しいクライアントを登録する場合、Integration Server は自動的に [認証コードの譲渡] チェックボックスを選択します。
- 認証コードの譲渡は Integration Server がサポートする 4 つの OAuth 譲渡タイプの中で最も安全です。
- [認証コードの譲渡] または [暗黙的譲渡] チェックボックスを選択する場合、クライアントに対して少なくとも 1 つの [リダイレクト URI] を入力する必要があります。
- クライアントクレデンシャルの譲渡を使用するクライアントを登録した後で、クライアントに必要なリソースにアクセスする権限を与える必要があります。このことは、ほかの譲渡と異なります。ほかの譲渡では、リソース所有者が譲渡フローに関与しているため Integration Server は所有者を知っています。具体的には、リソースにアクセスするためにアクセストークンが使用されている場合、Integration Server は所有者がリソースにアクセスする権限を持っていることを確認します。クライアントクレデンシャルの譲渡では、Integration Server は所有者を知りません。このため、クライアントがリソースにアクセスする権限を持っている必要があります。クライアントの登録後、クライアントが必要とするアクセスコントロールリスト (ACL) に属する 1 つ以上のユーザグループに、クライアントを追加する必要があります。Integration Server がクライアントに対して生成したクライアント ID は、ユーザ名として [**セキュリティ**] > {ユーザ管理] ページに表示されます。

注:
Integration Server の 10.3 よりも前のバージョンでは、クライアントの登録には、許可される譲渡タイプの指定は要求されませんでした。正確には、すべての登録されたクライアントは、認証コードの譲渡タイプおよび暗黙的譲渡タイプの使用を許可されていました。Integration Server バージョン 10.3 より、すべての新しいクライアントは少なくとも 1 つの譲渡タイプを指定する必要があります。Integration Server 10.3 に移行したクライアントは、引き続き認証コードの譲渡タイプおよび暗黙的譲渡タイプを使用できます。

クライアントを認証サーバに登録するには、以下の手順に従います。

クライアントを登録するには

1. 認証サーバとして定義された Integration Server の Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [OAuth] に移動します。
3. [**クライアント登録**] をクリックします。
4. [**クライアントの登録**] をクリックします。
5. [**クライアント設定**] で、各フィールドに以下のように入力します。

フィールド	説明
[名前]	クライアントの名前を指定します。 名前に以下の文字は使用できません。 & ( ) \ ; , / " : ' < > **注:** 同じ [**名前**] と [**バージョン**] の組み合わせでクライアントを作成することはできません。
[バージョン]	クライアントのバージョン番号を指定します。 バージョンに以下の文字は使用できません。 & ( ) \ ; , / " : ' < > **注:** 同じ [**名前**] と [**バージョン**] の組み合わせでクライアントを作成することはできません。
[タイプ]	認証サーバとの通信機能に応じて、クライアントのタイプを指定します。

指定する値	条件
[機密]	クライアントは、セキュアなクライアント認証を維持できます。クライアントタイプを [**機密**] として選択した場合、Integration Server によってクライアントシークレットが生成されます。クライアントが OAuth サービスに対して要求を行うと、このクライアントシークレットが Integration Server によって要求されます。OAuth セッションが、認証コードの譲渡タイプ、リソース所有者のパスワードクレデンシャルの譲渡タイプ、およびクライアントクレデンシャルのの譲渡タイプを使用する場合は、[**機密**] を指定します。
[公開]	クライアントは、セキュアなクライアント認証を維持できません。OAuth セッションで暗黙的譲渡タイプが使用される場合は、[**公開**]

フィールド	説明
	を指定します。詳細については、644 ページの「暗黙的譲渡」を参照してください。
説明	オプション。クライアントの説明を指定します。
[リダイレクト URI]	譲渡プロセスでリソース所有者のブラウザをリダイレクトするために認証サーバによって使用される URI を指定します。 一度に複数の URI を追加するには、1 行に 1 URI ずつ複数の行を指定します。行を分割するには Enter キーを押します。 [認証コードの譲渡] または [暗黙的譲渡] タイプを選択する場合、クライアントに対して少なくとも 1 つの [リダイレクト URI] を入力する必要があります。
許可される譲渡	クライアントが Integration Server で使用できる OAuth 譲渡タイプを指定します。次の 1 つ以上のアイコンを選択します。 ■ **認証コードの譲渡** ■ **暗黙的譲渡** ■ **クライアントクレデンシャルの譲渡** ■ **リソース所有者のパスワードクレデンシャルの譲渡** デフォルトは、最も安全な譲渡タイプである [**認証コードの譲渡**] です。 [**認証コードの譲渡**] または [**暗黙的譲渡**] を選択する場合、少なくとも 1 つの [リダイレクト URI] をクライアントに対して入力する必要があります。 機密クライアントだけが、クライアントクレデンシャルおよびリソース所有者のパスワードクレデンシャルの譲渡を使用できます。公開クライアントに対しては、これらの譲渡を選択できません。クライアントが許可されていない譲渡を使用してアクセストークンを要求した場合、Integration Server は要求を拒否し、unauthorized_client OAuth エラーを返します。

6. [**トークン**] で、クライアントに発行するトークンについて以下の情報を指定します。

フィールド	説明
[有効期間]	アクセストークンが有効な時間の長さ (秒単位) を指定します。 **選択項目** **目的**

フィールド	説明	
	[OAuth グローバル設定を使用する]	[OAuth] ページの [**アクセストークン有効期間**] フィールドで指定した設定を使用します。この設定は、山カッコ内に表示されます。[**アクセストークン有効期間**] フィールドの詳細については、652 ページの「OAuth 設定の設定」を参照してください。
	[無期限]	アクセストークンが期限切れにならないことを示します。
	[有効期限]	特定の間隔を指定します。アクセストークンが有効な秒数を、表示されているフィールドに入力します。最大値は 2147483647 です。デフォルトは 3600 です。
[リフレッシュ回数]	アクセストークンをリフレッシュできる回数を指定します。 **注:** 認証コードの譲渡フローおよびリソース所有者のパスワードクレデンシャル (ROPC) の譲渡フローを使用する場合のみ、トークンをリフレッシュできます。	
	選択項目	**目的**
	[無制限]	アクセストークンを回数の制限なくリフレッシュできます。
	制限	Integration Server でアクセストークンをリフレッシュできる回数を指定します。 0 より大きい値を指定した場合、指定した回数アクセストークンをリフレッシュできるように、Integration Server によってリフレッシュトークンが発行されます。アクセストークンが期限切れになったら、クライアントは pub.oauth:getToken サービスを使用して、トークンリフレッシュ要求を認証サーバにサブミットできます。 0 を指定するか [**制限**] フィールドを空のままにした場合、Integration Server によってリフレッシュトークンは発行されません。 最大値は 2147483647 です。デフォルトは 0 です。

7. [**変更内容の保存**] をクリックします。

Integration Server によってクライアント ID が生成されます。[**タイプ**] フィールドで [**機密**] を指定した場合は、Integration Server によってクライアントシークレットも生成されます。クライアントが OAuth サービスを呼び出すと、Integration Server によってクライアント ID、クライアントシークレット、またはこれら両方が要求されます。

メモ:

- 機密クライアントが登録された場合は、対応する Integration Server ユーザアカウントが作成されます。ユーザ名はクライアント ID であり、パスワードはクライアントシークレットです。既存のクライアントが機密から公開に、またはその逆に変更された場合は、対応するユーザアカウントが作成または削除されます。
- Integration Server ユーザアカウントに対して自動的に生成されるパスワードは、パスワード制限に適合しないことがあります。ただし、Integration Server で OAuth クライアントに対して自動的に生成されるパスワードは、セキュアな 36 文字の疑似ランダム値です。
- 機密 OAuth クライアントがクラスタ化された Integration Server に作成された場合、クライアントのユーザアカウントはローカルファイルシステムに保存されますが、クラスタの他のサーバには保存されません。クライアントがクラスタの別のサーバの OAuth トークンエンドポイントに要求を送信した場合、ユーザアカウントは他方のサーバのクライアントに対して動的に作成されます。

クライアントの有効化および無効化

登録されたクライアントに発行されたすべてのアクセストークンについて、リソースへのアクセスを一時的に無効にする場合は、そのクライアントを無効にすることができます。クライアントを無効にすると、イントロスペクションエンドポイント (pub.oauth:introspectToken) はクライアントに発行されたすべてのトークンに対して「active=false」を返します。そのため、リソースサーバは、無効になったクライアントのトークンの 1 つを使用する要求へのアクセスを拒否します。

注:
watt.server.oauth.disableClient.disableTokens サーバ設定パラメータでは、アクセストークンがアクティブかどうかを判断するときにクライアントアカウントが無効か有効かを考慮に入れるかどうかを OAuth イントロスペクションエンドポイント pub.oauth:introspectToken service で制御します。watt.server.oauth.disableClient.disableTokens を「true」に設定した場合、pub.oauth:introspectToken サービスでは、トークン発行先のクライアントアカウントが無効であると、トークンは非アクティブであると見なします。「false」に設定した場合、pub.oauth:introspectToken サービスでは、アクセストークンの評価時に、アクセストークン発行先のクライアントアカウントの有効/無効状態を考慮に入れません。
watt.server.oauth.disableClient.disableTokens の詳細については、次を参照してください:
1136 ページの「watt.server.」

登録されたクライアントを有効または無効にするには、以下の手順に従います。

クライアントを有効または無効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [OAuth] に移動します。
3. [**クライアント登録**] をクリックします。
4. [**登録されたクライアント**] リストの [**アクティブ**] 列で、以下のいずれかを選択します。

クリック	[宛先]
いいえ	クライアントを有効にします。
はい	クライアントを無効にします。

5. 登録されたクライアントを有効または無効にするかどうかの確認を求められたら、[**OK**] をクリックします。

クライアントの編集

登録されたクライアントを編集するには、以下の手順に従います。

クライアントを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**クライアント登録**] をクリックします。
4. [**登録されたクライアント**] で、編集するクライアントの [**アプリケーション名**] または [**クライアント ID**] をクリックします。
5. クライアントの情報を更新します。

> **注:**
> [**ID**] フィールドまたは [**秘密**] フィールドに表示されているデータは編集できません。Integration Server Administrator は [**秘密**] フィールドの内容をマスクします。シークレットを表示するには、[**シークレットの開示**] をクリックします。もう一度シークレットをマスクするには、[**シークレット隠す**] をクリックします。

6. [**変更内容の保存**] をクリックします。

クライアントの削除

登録されたクライアントを削除するには、以下の手順に従います。

> **重要:**

クライアントを削除すると、Integration Server によってすべてのアクセスも削除され、そのクライアントのトークンはリフレッシュされます。

クライアントを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**クライアント登録**] をクリックします。
4. [**登録されたクライアント**] リストでクライアントを見つけ、[**削除**] 列で ✕ アイコンをクリックします。
5. 登録されたクライアントを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

スコープの定義

スコープは、クライアントがリソース所有者に代わってアクセスできるリソースです。スコープは、名前と 1 つ以上の Integration Server フォルダまたはサービス、あるいはこれら両方で構成されます。スコープに対してアクセスが譲渡された場合は、そのスコープ内のすべてのフォルダおよびサービスに対してアクセスが譲渡されます。スコープを登録されたクライアントにマッピングして、Integration Server でクライアントがアクセスできるネームスペースリソースを示します。クライアントへのスコープのマッピングの詳細については、665 ページの「スコープとクライアントの関連付け」を参照してください。

認証サーバに対して要求が行われると、クライアントにスコープが定義されているかどうかが Integration Server によって検証されます。クライアントは、スコープに指定されたネームスペースリソースにのみアクセスできます。要求されたスコープが定義されていない場合は、スコープが無効であることを示すエラーが Integration Server によって返されます。

スコープの追加

スコープを追加するには、以下の手順に従います。スコープによって、クライアントがアクセスする権限を持つサービスおよびフォルダが定義されます。

スコープを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**スコープの管理**] をクリックします。
4. [**スコープの追加**] をクリックします。

5. [**スコープの設定**] で、以下の情報を指定します。

フィールド	説明
[名前]	スコープの一意の名前を指定します。スコープ名は、33～126 文字の ASCII 文字コードである必要があり、以下の文字は使用できません。 \ , "
[説明]	スコープの説明。 **注:** Integration Server が Swagger ドキュメントに基づいて REST API 記述子の作成中に自動的にスコープを生成する場合、Integration Server はデフォルトのスコープの説明を XX として追加します。このスコープの説明は変更できます。ただし、Integration Server で同じ Swagger ドキュメントを使用して RAD を再生成した場合、その説明は XX として上書きされます。
[フォルダとサービス]	クライアントがリソース所有者に代わってアクセスできるフォルダおよび個別のサービスのリストを指定します。 一度に複数のフォルダまたはサービスを追加するには、1 行に 1 フォルダまたは 1 サービスずつ複数の行を指定します。行を分割するには Enter キーを押します。
[URL テンプレート]	スコープを使用してセキュリティで保護する REST V2 リソースの URL テンプレートを指定します。 **注:** Integration Server が Swagger ドキュメントに基づいて REST API 記述子の作成中に自動的に URL を生成する場合、Integration Server はデフォルトの URL テンプレートを *<HTTP method>_<REST API descriptor namespace>/<URL template of resource>* として追加します。たとえば、次のように入力します。 `GET_RadNamesSpace/employee` このテンプレートは変更できます。ただし、Integration Server で同じ Swagger ドキュメントを使用して RAD を再生成した場合、そのテンプレートはデフォルトの URL テンプレートとして上書きされます。

6. [**変更内容の保存**] をクリックします。

スコープの編集

スコープを編集するには、以下の手順に従います。

スコープを編集するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**スコープの管理**] をクリックします。
4. [**スコープの設定**] で、編集するスコープの [**名前**] をクリックします。
5. スコープの情報を更新します。

   注:
   スコープの [**名前**] フィールドは変更できません。

6. [**変更内容の保存**] をクリックします。

スコープの削除

スコープを削除するには、以下の手順に従います。

注:
クライアントによって使用されているスコープは削除できません。スコープがクライアントによって使用されているかどうかを確認するには、667 ページの「クライアントとスコープ間の関連付けの表示」を参照してください。

スコープを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**スコープの管理**] をクリックします。
4. [**定義されたスコープ**] リストでスコープを見つけ、[**削除**] 列で ✕ アイコンをクリックします。
5. スコープを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

スコープとクライアントの関連付け

スコープとクライアントの関連付けは、[スコープをクライアントに関連付け] ページで管理します。

[スコープをクライアントに関連付け] ページは、以下の 2 つの領域に分かれています。

- [**スコープ**] 領域 (左側) では、スコープをクライアントに関連付けることができます。複数のクライアントを 1 つのスコープに関連付ける場合は、[**スコープ**] 領域を使用して関連付けを作成します。この領域には、スコープに対して複数の関連付けられたクライアントを一度に追加または削除できるという利点があります。
- [**クライアント**] 領域 (右側) では、クライアントをスコープに関連付けることができます。複数のスコープを 1 つのクライアントに関連付ける場合は、[**クライアント**] 領域を使用して関連付けを作成します。この領域には、クライアントに対して複数の関連付けられたスコープを一度に追加または削除できるという利点があります。

注:
関連付ける前に、スコープおよびクライアントを定義する必要があります。 クライアントの登録の詳細については、655 ページの「クライアントの定義」を参照してください。スコープの追加の詳細については、662 ページの「スコープの定義」を参照してください。

クライアントとスコープ間の関連付けの追加

スコープとクライアント間の関連付けを追加するには、以下の手順に従います。

スコープとクライアント間の関連付けを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**スコープの管理**] をクリックします。
4. [**スコープをクライアントに関連付け**] をクリックします。
5. 1 つ以上のクライアントを特定のスコープに関連付けるには、次の作業を行ってください。
   a. [**スコープ**] の [**スコープを選択**] リストで、1 つ以上のクライアントへの関連付けを追加するスコープを選択します。

      [スコープをクライアントに関連付け] ページの [**スコープ**] 領域 (左側) には、2 つのリストがあります。[**スコープに関連付けされたクライアント**] は、選択したスコープに現在関連付けられているクライアントのリストです。[**残りのクライアント**] は、現在は選択したスコープ内にないクライアントのリストです。

   b. [**残りのクライアント**] リストで、スコープに追加するクライアントを選択して (複数選択可) ハイライトします。

現在選択しているクライアントを解除せずにさらにクライアントを追加するには、Ctrl キーを押しながら、追加するクライアントをクリックします。クライアントの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

c. スコープに追加するクライアントをすべて選択したら、 をクリックします。Integration Server によって、選択したクライアントは [**スコープに関連付けされたクライアント**] リストに移動されます。

6. 1 つ以上のスコープを特定のクライアントに関連付けるには、次の作業を行ってください。

a. [**クライアント**] の [**クライアントを選択**] リストで、1 つ以上のスコープへの関連付けを追加するクライアントを選択します。

[スコープをクライアントに関連付け] ページの [**クライアント**] 領域 (右側) には、2 つのリストがあります。[**このクライアント内のスコープ**] は、選択したクライアントに現在関連付けられているスコープのリストです。[**残りのスコープ**] は、現在は選択したスコープ内にないクライアントのリストです。

b. [**残りのスコープ**] リストで、クライアントに追加するスコープを選択して (複数選択可) ハイライトします。

現在選択しているスコープを解除せずにさらにスコープを追加するには、Ctrl キーを押しながら、追加するスコープをクリックします。スコープの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

c. クライアントに追加するスコープをすべて選択したら、 をクリックします。Integration Server によって、選択したスコープは [**このクライアント内のスコープ**] リストに移動されます。

7. [**変更内容の保存**] をクリックします。

クライアントとスコープの関連付けの削除

クライアントとスコープの関連付けを削除するには、以下の手順に従います。

注:
スコープとクライアント間の関連付けを削除しても、クライアントまたはスコープはシステムから削除されず、既に発行されているアクセストークンにも影響しません。クライアントの削除の詳細については、661 ページの「クライアントの削除」を参照してください。スコープの削除の詳細については、664 ページの「スコープの削除」を参照してください。

クライアントとスコープの関連付けを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [OAuth] に移動します。

3. [**スコープの管理**] をクリックします。

4. [**スコープをクライアントに関連付け**] をクリックします。

5. 1 つ以上のクライアントの関連を特定のスコープから削除するには、次の作業を行ってください。

   a. [**スコープ**] の [**スコープを選択**] リストで、1 つ以上のクライアントから関連付けを削除するスコープを選択します。

   b. [**スコープに関連付けされたクライアント**] リストで、スコープから削除するクライアントを選択して (複数選択可) ハイライトします。

      現在選択しているクライアントを解除せずにさらにクライアントを追加するには、Ctrl キーを押しながら、追加するクライアントをクリックします。クライアントの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

   c. スコープから削除するクライアントをすべて選択したら、 をクリックします。Integration Server によって、選択したクライアントは [**残りのクライアント**] リストに移動されます。

6. 1 つ以上のスコープの関連を特定のスコープから削除するには、次の作業を行ってください。

   a. [**クライアント**] の [**クライアントを選択**] リストで、1 つ以上のスコープから関連付けを削除するクライアントを選択します。

   b. [**このクライアント内のスコープ**] リストで、クライアントから削除するスコープを選択して (複数選択可) ハイライトします。

      現在選択しているスコープを解除せずにさらにスコープを追加するには、Ctrl キーを押しながら、追加するスコープをクリックします。スコープの選択を解除するには、Ctrl キーを押しながら、現在選択されているエントリをクリックします。

   c. クライアントから削除するスコープをすべて選択したら、 をクリックします。Integration Server によって、選択したスコープは [**残りのスコープ**] リストに移動されます。

7. [**変更内容の保存**] をクリックします。

クライアントとスコープ間の関連付けの表示

スコープとクライアント間の関連付けを表示するには、以下の手順に従います。

クライアントとスコープ間の関連付けを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [OAuth] に移動します。

3. [**スコープの管理**] をクリックします。

4. [**スコープをクライアントに関連付け**] をクリックします。

5. 次のいずれかの手順に従います。

   - 特定のスコープに関連付けれられたクライアントを表示するには、[**スコープ**] の [**スコープを選択**] リストで、関連付けられているクライアントを表示するスコープを選択します。
   - 特定のクライアントに関連付けれられたスコープを表示するには、[**クライアント**] の [**クライアントを選択**] リストで、関連付けられているスコープを表示するクライアントを選択します。

トークンの管理

認証サーバによって発行されたアクティブなトークンを表示および削除するには、[トークン] ページを使用します。これらのトークンは、クライアントアプリケーションによって、Integration Server (リソースサーバ) 上のリソースにアクセスするために使用されます。トークンを削除すると、クライアントアプリケーションは、リソース所有者が所有するリソースにアクセスできなくなります。

また、WmPublic パッケージに含まれる組み込みサービスを使用し、認証サーバとして機能する Integration Server が発行するトークンを取り消し、期限切れのアクセストークンを削除することもできます。

トークンの表示

認証サーバによって発行されたアクティブなトークンを表示するには、以下の手順に従います。

トークンを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**セキュリティ**] > [OAuth] に移動します。

3. [**トークン**] をクリックします。[**トークン**] には、Integration Server によって発行されたアクティブなアクセストークンと期限切れのアクセストークンが Integration Server Administrator によってリストされています。

トークンの削除

[トークン] ページを使用して、アクティブなトークンをすべて削除できます。アクティブなトークンのリストからトークンを削除すると、Integration Server によってアクセストークンとリフレッシュトークンの両方が削除されます。トークンを削除する場合は、以下の点に留意してください。

- トークンの削除後、同じクライアントアプリケーションを使用する他のリソース所有者は、クライアントアプリケーションの使用を続行できます。特定のクライアントアプリケーションでリソースにアクセスできなくするには、クライアントアプリケーションを無効化または削除します。クライアントアプリケーションの削除の詳細については、661 ページの「クライアントの削除」を参照してください。
- トークンが削除されると、そのアクセストークンまたはリフレッシュトークンを使用するクライアントからの要求は Integration Server によって拒否されます。

トークンを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**トークン**] をクリックします。
4. [**トークン**] リストでトークンを見つけ、[**削除**] 列で ✕ アイコンをクリックします。
5. トークンを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

トークンの取り消し

WmPublic パッケージの pub.oauth:revokeToken サービスを使用して、Integration Server が発行したトークンを取り消すことができます。pub.oauth:revokeToken サービスは指定されたトークンに関連するすべてのトークンを取り消します。トークンがアクセストークンである場合は、サービスは関連するリフレッシュトークンも取り消します。トークンがリフレッシュトークンである場合は、サービスは関連するアクセストークンも取り消します。

pub.oauth:revokeToken サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

期限切れのアクセストークンの削除

WmPublic パッケージの pub.oauth:removeExpiredAccessTokens サービスを使用して、データベースから期限切れの OAuth アクセストークンを削除できます。
pub.oauth:removeExpiredAccessTokens サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

承認ページのカスタマイズ

承認ページは、クライアントによってプライベートリソースへのアクセス要求がサブミットされた後、認証サーバからリソース所有者に送信される HTML ページです。リソース所有者は、このページを使用して要求を承認または拒否します。

OAuth 承認デフォルトページ

承認ページのタイトル、ロゴ、見出し、フッターをカスタマイズするには、watt.server.oauth.approvalpage のプロパティを使用します。これらのプロパティの使用の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

リソースサーバとしての Integration Server の使用について

クライアントは、保護されたリソースへのアクセスを要求してアクセストークンを受け取ると、そのアクセストークンをリソースサーバに提示できます。Integration Server がリソースサーバである場合、クライアントは承認要求ヘッダーフィールドを使用してアクセストークンを提示できます。クライアントは、ベアラー認証スキームを使用して承認要求ヘッダーフィールド内のアクセストークンを送信し、アクセストークンをサブミットできます。次に例を示します。

```
GET /invoke/folder/svc HTTP/1.1
Host: your-company.com:5555
Authorization: Bearer access_token_id
```

pub.client:http サービスを使用している場合は、auth ヘッダーフィールドを使用してトークンを送信できます。このサービスの使用の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

ベアラー認証スキームの詳細については、OAuth 2.0 Authorization Framework Bearer Token Usage 仕様を参照してください。

外部認証サーバの使用

Integration Server がリソースサーバである場合、認証サーバを指定する必要があります。認証サーバとして Integration Server を使用する代わりに、サードパーティのサーバを認証サーバとして使用できます。このようにすると、Integration Server では、サードパーティベンダーが RFC 7662 (OAuth 2.0 Token Introspection) をサポートするサードパーティの OAuth 2.0 認証サーバによって作成された OAuth ベアラートークンを使用できます。

外部認証サーバを使用するには、次の作業を行う必要があります。

- サードパーティの認証サーバを設定します。次の設定が含まれますが、これらに制限されるものではありません。
  - Integration Server で認証サーバのイントロスペクションエンドポイントに使用するクライアントアカウントを作成します。

    client_id と client_secret の値を書き留めます。この情報は、Integration Server リソースサーバの外部認証サーバエイリアスを定義するときに使用します。

    イントロスペクションエンドポイントの URL を書き留めます。この情報は、Integration Server リソースサーバで外部認証サーバエイリアスを定義するときに使用します。
  - 1 つまたは複数の OAuth スコープを作成します。これらは Integration Server リソースサーバで作成する OAuth スコープの名前に一致させる必要があります。

  OAuth 2.0 認証サーバの作成と設定の詳細については、ベンダーが提供するマニュアルを参照してください。
- 認証サーバにエイリアスを設定します。詳細については、671 ページの「外部認証サーバエイリアスの作成」を参照してください。
- [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**認証サーバ**] の値として外部認証サーバを選択します。

現在、Integration Server は次のような RFC 7662 (OAuth 2.0 Token Introspection) をサポートする外部認証サーバとともに使用できます。

- Okta
- Ping Identity

注:
外部認証サーバからのアクセストークンが拒否された場合、拒否に関する情報がサーバログに書き込まれます。サーバログのこれらのメッセージを表示するには、OAuth ログ機能 (0010) をデバッグに設定します。

外部認証サーバエイリアスの作成

リソースサーバとして機能する Integration Server で認証サーバとしてサードパーティサーバを使用する場合、外部認証サーバエイリアスを作成する必要があります。

外部認証サーバエイリアスを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [**外部認証サーバの追加**] をクリックします。
4. [外部認証サーバの設定] の下で、次の情報を指定します。

フィールド	説明
[名前]	外部認証サーバのエイリアス。?[ ] / \ = + < > : ; " , * | ^ @ の文字は使用できません。
[イントロスペクションエンドポイント]	外部認証サーバのイントロスペクションエンドポイントの URL。Integration Server ではイントロスペクションエンドポイントを使用して、クライアント要求で使用されるアクセストークンが現在アクティブであることを特定します。
[クライアント ID]	Integration Server で外部認証サーバのイントロスペクションエンドポイントに要求を送信するときに使用するユーザアカウントの ID。
[クライアントシークレット]	Integration Server で外部認証サーバのイントロスペクションエンドポイントに要求を送信するときに使用するユーザアカウントのパスワード。
[ユーザ]	Integration Server でクライアント要求の実行に使用する Integration Server ユーザアカウント。クライアントがサービスを要求している場合は、これが、Integration Server がサービスの実行に使用するユーザアカウントです。サービスの実行は Integration Server がイントロスペクションエンドポイントを呼び出した後に行われます。クライアントがファイルを要求している場合、これが、Integration Server がファイルへのアクセスに使用するユーザアカウントです。 [**ユーザ**] の値が使用されるのは、外部認証サーバのイントロスペクションエンドポイントによってアクセストークンが現在アクティブであることが示される場合だけです。 アイコンをクリックしてユーザを検索して選択します。ローカルディレクトリまたはセントラルディレクトリからユーザを選択できます。
[キーストアエイリアス (オプション)]	Integration Server で相互 (双方向) SSL ハンドシェイク間に外部認証サーバに送信されるデジタル認証を保持する Integration Server のキーストアのエイリアス。キーストアエイリアスの選択が必要となるのは、外部認証サーバのクライアントアカウントが相互 (双方向) SSL を使用するように設定される場合だけです。
[キーエイリアス (オプション)]	Integration Server で相互 (双方向) SSL ハンドシェイク間に外部認証サーバに送信される Integration Server の秘密鍵および関連するデジタル認証のエイリアス。キーエイリアスの選択が必要となるのは、外部認証サーバのクライアントアカウントが相互 (双方向) SSL を使用するように設定される場合だけです。
[トラストストアエイリアス (オプション)]	外部認証サーバの認証用の認証局 (CA) を保持する Integration Server のトラストストアのエイリアス。以下のすべての条件に当てはまる場合にのみ、トラストストアエイリアスを選択する必要があります。 ■ 外部認証サーバのクライアントアカウントで相互 (双方向) SSL の使用が設定されている。

フィールド	説明
	■ 認証サーバの認証局の認証が、Integration Server が実行されている JVM によって信用されるよく知られている一連の認証局の中にない。 ■ watt.security.cert.wmChainVarifier.trustByDefault property プロパティが「false」に設定されている。
[デフォルトのスコープ]	外部認証サーバのイントロスペクションエンドポイントからの応答でスコープが明示的に指定されていない場合に有効になるデフォルトのスコープ。外部認証サーバに定義されたとおりにこのフィールドにスコープ値を入力します。 外部認証サーバのイントロスペクションエンドポイントからの応答がスコープ値を返さず、[デフォルトのスコープ] が指定されていない場合、Integration Server では認証サーバのアクセストークンを含む要求はスコープ外であると見なし、要求を拒否して 401 応答を発行します。

5. [**Save**] をクリックします。

   Integration Server Administrator の [**セキュリティ**] > [OAuth] ページの [外部認証サーバ] に新しい外部認証サーバが表示されます。

外部認証サーバの削除

リソースサーバとして機能する Integration Server の認証サーバとしてサードパーティサーバを使用する必要がなくなった場合、外部認証サーバエイリアスを削除できます。

外部認証サーバエイリアスを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [OAuth] に移動します。
3. [外部認証サーバ] リストで外部エイリアスを探し、[**削除**] 列で ✕ アイコンをクリックします。
4. エイリアスを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

36 セントラルユーザディレクトリまたは LDAP の設定

- 操作を開始する前に 676
- Integration Server が外部定義されたユーザおよびグループを扱う方法の概要 676
- セントラルユーザ管理の設定 678
- LDAP の使用の概要 680
- LDAP を使用する際のサーバの設定 681
- ユーザアカウントとグループに関する考慮事項 688
- 外部ユーザへの Administrator 特権の付与について 690
- 外部ユーザへの Developer 特権の付与 691
- 外部ユーザへのサービスとファイルに対するアクセス特権の付与 692

操作を開始する前に

ここでは、Integration Server 内部で定義されたユーザ情報およびグループ情報の代わりに、外部ディレクトリを使用してクライアントを認証する方法を説明します。内部定義されたユーザおよびグループの使用方法の詳細については、101 ページの「ユーザとグループの管理」を参照してください。また、サイトでユーザとグループの情報として次に示す外部ディレクトリのいずれかを使用している場合は、外部ディレクトリの情報にアクセスするように Integration Server を設定することもできます。

- 共通ディレクトリサービスおよび My webMethods Server を介したセントラルユーザ管理
- LDAP (Lightweight Directory Access Protocol)

Integration Server を設定し、外部ディレクトリとしてセントラルユーザディレクトリまたは LDAP のいずれか 1 つだけを使用できます。

重要:
ユーザ環境内で My webMethods Server を使用している場合は、セントラルユーザ管理ディレクトリを使用するように Integration Server を設定することをお勧めします。LDAP サーバを使用したい場合、My webMethods Server を使用しているのであれば、My webMethods Server を使用して LDAP を設定することをお勧めします。My webMethods Server を使用していない場合のみ、LDAP サーバと対話できるように Integration Server を直接設定してください。

ここで先に進む前に、Integration Server がユーザ情報およびグループ情報をどのように使用するかを理解しておくと便利です。詳細については、次の章を参照してください。

- 101 ページの「ユーザとグループの管理」
- 106 ページの「Administrator ユーザの追加」
- 107 ページの「Developer ユーザの追加」
- 534 ページの「ACL によるリソースへのアクセス制御」
- 550 ページの「基本認証」

Integration Server が外部定義されたユーザおよびグループを扱う方法の概要

ここでは、セントラルユーザディレクトリまたは LDAP に定義されているユーザおよびグループを、Integration Server がいつどのように扱うかを具体的に次の観点から説明します。

- 外部定義されたユーザおよびグループを Integration Server でどのように使用できるか。
- 外部定義されたユーザおよびグループに関する情報に Integration Server がいつアクセスするか。
- 外部定義されたグループおよび役割に属するユーザを Integration Server がどのように認証するか。

外部定義されたユーザおよびグループをサーバがどのように使用するか

Integration Server は、内部定義されたユーザ情報やグループ情報と同様に、外部定義された情報を次の用途に使用できます。

- ユーザ名とパスワードを使用したクライアントの認証
- Integration Server を設定および管理できるユーザの制御
- Software AG Designer を使用してサービスを作成、変更および削除できるユーザの制御
- Integration Server 内の使用可能なサービスおよびファイルへのアクセスの制御

外部定義された情報は、ACL に置き換わるものではありません。サービスやファイルへのアクセスを制御するには、従来どおり、ACL を設定して、特定のサービスやファイルへのアクセスを許可または拒否するグループを識別する必要があります。ただし、外部定義されたグループを ACL に割り当てることは可能です。

セントラルユーザ管理または LDAP ディレクトリを使用するようにサーバを設定した場合、外部定義のユーザおよびグループは [セキュリティ] > [**ユーザ管理**] ページには表示されません。ただし、外部グループが Integration Server の ACL にマッピングされている場合、そのグループは [セキュリティ] > [**ACL**] ページに表示されます。

注:
外部定義されたユーザグループの名前にアポストロフィ (') が含まれている場合、Integration Server でそのグループは [**セキュリティ**] > [ACL] ページに表示されません。

サーバが外部定義情報にアクセスする場合

Integration Server では、外部定義情報を取得して、クライアントを認証し、ACL がアクションを許可するか拒否するかを判断します。

注:
Integration Server がセントラルユーザディレクトリまたは LDAP ディレクトリにアクセスする必要があるクライアント要求の場合は、その必要がないクライアント要求よりも時間がかかることがあります。

Integration Server で外部定義のクライアントを認証する方法

Integration Server がユーザ名とパスワードを使用してクライアントを認証する際には、Integration Server は最初に内部定義された情報を使用して、ユーザ名とパスワードを探します。Integration Server が指定されたユーザ名に相当する内部定義のユーザアカウントを見つけた場合、Integration Server は内部定義された情報を使用して、指定されたクライアントを認証します。提供されたパスワードが正しければ、Integration Server は要求の処理を実行します。提供されたパスワードが誤っていれば、Integration Server は要求を拒否します。

提供されたユーザ名に相当する内部定義のユーザアカウントをIntegration Server が見つけられなかった場合、Integration Server は外部ディレクトリ (セントラルユーザディレクトリまたは

LDAP) にアクセスして、指定されたクライアントに相当するユーザ名情報とパスワード情報を取得します。Integration Server が外部定義されたユーザアカウントを見つけた場合、Integration Server は外部定義された情報を使用して、指定されたクライアントを認証します。たとえば、ユーザアカウントが My webMethods Server ユーザディレクトリで定義されている場合、Integration Server は My webMethods Server データベースで定義された情報を使用してクライアントを認証します。指定されたパスワードが正しければ、サーバは要求の処理を実行します。指定されたパスワードが誤っていれば、サーバは要求を拒否します。

注:
パスワードがセントラルユーザまたは LDAP 以外のケルベロスなどの外部認証システムに含まれている場合、プラグ接続可能な独自モジュールを作成してその情報を取得する必要があります。プラグ接続可能なモジュールをセットアップする方法については、561 ページの「JAAS を使用した認証のカスタマイズ」を参照してください。

指定されたユーザに相当する内部定義のユーザアカウントも外部定義のユーザアカウントも見つからなかった場合、サーバは要求を拒否します。

ユーザがユーザ名やパスワードを指定しなかった場合、サーバは内部定義された Default ユーザアカウントを使用します。

注:
ユーザは、 Integration Server の特定のインスタンスに対してローカルで、そして Integration Server インスタンスが利用可能な外部ディレクトリサービスで定義できます。両方で定義されたユーザが Integration Server Administrator にログインすると、Integration Server はローカルで定義された特権を使用してユーザを認証します。ユーザが外部ディレクトリサービスで異なる特権を定義されている場合、これらの特権は無視されます。特権の無視は、Integration Server がローカルユーザリストを最初にチェックするために生じます。指定されたユーザ名が存在してパスワードが正しい場合、Integration Server はそのユーザアカウントについて外部ディレクトリをチェックしません。ユーザがローカルおよび外部ディレクトリサービスで定義されている場合は、ユーザがローカルおよび外部で同じ特権を持っていることを確認する必要があります。

セントラルユーザ管理の設定

セントラルユーザ管理とは、webMethods 製品のユーザに関する情報を、1 つの場所で保存および管理することを意味します。Integration Server Administrator を使用して、セントラルディレクトリ内のユーザに、Integration Server の機能やサービスへのアクセス権を付与できます。たとえば、My webMethods Server の役割またはグループを ACL に割り当てることができます。

ユーザが My webMethods インタフェースを通じて Integration Server または Trading Networks にアクセスする場合は、My webMethods Server 内でユーザを作成してから、Integration Server Administrator を使用して、必要な領域へのアクセス権をユーザに付与します。ユーザが LDAP などの外部ディレクトリに既に定義されている場合は、外部ディレクトリと対話するように My webMethods Server を設定できます。このように設定すると、認証および認可の要求は引き続き Integration Server によって行われます。ただし、サーバは My webMethods Server データベースで定義されたディレクトリ設定を使用して、外部ディレクトリへの接続を作成します。

My webMethods Server ユーザディレクトリなどの中心的な場所で定義されているユーザは、セントラルユーザと呼ばれることがあります。セントラルユーザディレクトリを使用して認証する

ための機能は、My webMethods Server の一部である共通ディレクトリサービスによって提供されます。

Integration Server がセントラルユーザ管理を使用するための要件

- My webMethods Server がインストールされ、外部データベースを使用できるように設定されている必要があります。
- Integration Server がスタンドアロンの My webMethods Server に接続する場合でも、My webMethods Server クラスタリングが設定される必要があります。これは、My webMethods Server と通信を行うために Integration Server が使用するクライアントサイド共通ディレクトリサービス API が、クラストノードとして機能するためです。My webMethods Server クラスタリングを有効にするために、少なくとも 1 回は My webMethods Server を起動する必要があります。

  注:
  My webMethods Server 10.0 以降、My webMethods Server はその JMS プロバイダとして Universal Messaging サーバを使用する必要があります。My webMethods Server は、クラスタ通信およびクラスタ同期のために JMS プロバイダを使用します。

- Integration Server に、My webMethods Server データベースコンポーネントを指し示す JDBC 接続プールが存在し、CentralUsers 機能エイリアスがその接続プールを指し示している必要があります。[設定] > [リソース] ページで、[MWS SAML リゾルバ URL] フィールドが My webMethods Server のホストおよびポートを指し示している必要があります。
- My webMethods Server のシングルサインオンは My webMethods Server のホストおよびポートを指し示すように設定されている必要があります。このプロパティの詳細については、556 ページの「MWS シングルサインオンのリソース設定」を参照してください。

注:
セントラルユーザ管理を使用するように Integration Server が正常に設定され、サーバログ機能 0024 User Manager のログレベルが Info 以上に設定されている場合、Integration Server は起動時に「[ISS.0024.0012I] セントラルユーザ管理の初期化が成功しました。」というメッセージをログに記録します。

後でセントラルユーザ管理を停止する場合は、157 ページの「Integration Server を外部 RDBMS のデータベースコンポーネントへ接続する」に記載されている手順に従いますが、[**関連付けられたプールエイリアス**] リストで [**CentralUsers**] 機能に [**なし**] を選択してから、Integration Server を再起動します。

注:
Integration Server は Anonymous ACL を自動更新して、My webMethods Server から ACL に My webMethods Users の役割を取り込みます。

共通ディレクトリサービスおよびクラスタリングのための My webMethods Server の設定の詳細については、『*Administering My webMethods Server*』を参照してください。

注:
SSL を使用した LDAP を使うようセントラルユーザ管理が設定されている場合、`custom_wrapper.conf` ファイルで `javax.net.ssl` プロパティを設定します。

My webMethods Server クエリー役割に関する考慮事項

My webMethods Server で定義されているすべての役割 (LDAP クエリー役割やデータベースクエリー役割など) を Integration Server で評価する必要がある場合、サービス呼び出しを処理するときに、Integration Server が予想外に遅くなったり終了したりすることがあります。この問題は、CAF アプリケーションから WS クライアントコネクタを使用して **Integration Server** サービスを呼び出すとき、特に、以下の条件に該当する場合に、発生する場合があります。

- My webMethods Server で定義されている LDAP クエリー役割およびデータベースクエリー役割がある。
- Common Users が Integration Server で有効である。
- サービスに ACL が割り当てられている。

これらの役割の評価は外部クエリーの実行に依存しており、Integration Server のパフォーマンスに影響する場合があります。

サーバ内の LDAP クエリー役割またはデータベースクエリー役割が、Integration Server によって watt.cds.skip.role.types 拡張設定を使用して評価されるようにするかどうかを制御できます。セントラルユーザが有効な Integration Server で、ACL 管理に LDAP クエリー役割またはデータベースクエリー役割が使用されていない場合は、クエリー役割評価機能を無効にすることを検討できます。

watt.cds.skip.role.types 拡張設定の使用の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

LDAP の使用の概要

ユーザ情報とグループ情報のためにサイトで LDAP を使用すると、外部ディレクトリからユーザ情報とグループ情報を取得するように Integration Server を設定することができます。複数の LDAP ディレクトリを同時に使用するように Integration Server を設定することができ、それによって、別の場所または別の組織のユーザに関するさまざまな LDAP ディレクトリで Integration Server を動作させることが可能です。さらに、ほかのディレクトリのバックアップ用としてディレクトリを使用するために、複数の LDAP ディレクトリを管理することもできます。

重要:
LDAP サーバを使用してユーザ情報を保存したい場合、My webMethods Server を使用しているのであれば、My webMethods Server を使用して LDAP を設定することをお勧めします。My webMethods Server を使用していない場合のみ、LDAP サーバと対話できるように Integration Server を直接設定してください。

LDAP は、ネットワーク上でリソースに関する情報を容易に共有できるように設計されたプロトコルです。一般的に、ログイン ID やパスワードなどのプロファイル情報を保存するために使用します。また、その他の情報も保存できます。Integration Server は LDAP を使用して外部認証を実行します。

既存の LDAP 情報を使用することの利点は、ユーザ情報とグループ情報の集中リポジトリを利用できることにあります。システム管理者は、1 箇所でユーザを追加および削除できます。ユーザは、webMethods アプリケーション用のパスワードを別に覚える必要はなく、他のアプリケー

ション用と同じユーザ名およびパスワードを使用することができます。外部ディレクトリに保存されたユーザまたはグループを管理するには、LDAP のツールを使用する必要があります。

LDAP およびキャッシングについて

LDAP の場合、Integration Server はアクセスされたユーザ情報をキャッシュするのでパフォーマンスが向上します。その情報がアクセスされずに 1 時間キャッシュに保持されたままの場合、または新しい要求のためにキャッシュスペースが必要な場合、Integration Server はキャッシュからその情報を削除します。

キャッシュにある情報を必要とする後続の要求をサーバが受信した場合、Integration Server は、外部ディレクトリにアクセスせずにキャッシュされている情報を使用します。

LDAP を使用する際のサーバの設定

LDAP を使用するようにサーバを設定するには、次の作業を実行する必要があります。

- LDAP プロトコルを使用するように Integration Server に通知します。
- そのユーザに対して Integration Server が使用する設定済み LDAP サーバを 1 台または複数定義します。
- LDAP プロバイダが SSL 対応の場合は、LDAP サーバとの安全な接続を確立するために必要な認証を含むトラストストアエイリアスをポイントするように [**セキュリティ**] > [認証] ページで [**JVM トラストストアエイリアス**] フィールドを設定できます。

  注:
  [**JVM トラストストアエイリアス**] フィールドの値は watt.server.ssl.trustStoreAlias パラメータの値として使用されます。[**JVM トラストストアエイリアス**] を使用する代わりに、watt.server.ssl.trustStoreAlias パラメータを設定することもできます。

Software AG では、外部ユーザの管理に追加の LDAP ディレクトリを使用するように Integration Server を設定する代わりに、セントラルユーザ管理を使用することをお勧めします。セントラルユーザ管理の詳細については、678 ページの「セントラルユーザ管理の設定」および『*Administering My webMethods Server*』を参照してください。

LDAP を外部プロバイダとして指定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**LDAP 設定**] をクリックします。
4. [**LDAP 設定の編集**] をクリックします。
5. [**プロバイダ**] の横で [**LDAP**] を選択します。

Integration Server によって、設定変更を確認するプロンプトが表示されます。[**OK**] をクリックします。Integration Server でセントラルユーザ管理に対応した設定が行われている場合は、LDAP を設定する前にこの設定を無効化しておく必要があります。セントラルユーザ管理の無効化の詳細については、678 ページの「セントラルユーザ管理の設定」を参照してください。

6. 次の情報を入力します。

フィールド	指定する値
[キャッシュサイズ (ユーザ数)]	Integration Server がユーザキャッシュに保持できる LDAP ユーザの最大数。デフォルトは 10 です。 制限に達した場合、Integration Server は使用されていない時間を基準にしてキャッシュから削除するユーザを選択します。そのため、アクティビティによってユーザがキャッシュに残る時間が延びます。 通例として、LDAP システムにはユーザ数の 5～10% に相当するキャッシュサイズを指定します。ただし、同時にログオンするセッションが少ない場合は、同時セッションの数と同じになるようにキャッシュサイズを設定します。
[クレデンシャルを廃棄するまでの時間 (分)]	LDAP ユーザのクレデンシャル (ユーザ ID とパスワード) をクレデンシャルキャッシュに保持できる、消去までの分単位の時間。デフォルトは 60 分です。 ユーザが最初にログインしようとするときに、Integration Server はユーザオブジェクトを作成し、ユーザのクレデンシャルを LDAP ディレクトリと照合します。Integration Server はこのクレデンシャルを保存するので、この後の認証要求は LDAP ディレクトリに対してではなくてキャッシュされているクレデンシャルに対して行われます。 セキュリティの理由から、キャッシュされたクレデンシャルが有効である時間を制御することができます。クレデンシャルは単方向ハッシュ関数を使用して保存され、キャッシュからは回復できないので安全です。キャッシュされているバージョンと一致しないクレデンシャルを使用してユーザがログインしようとすると、Integration Server はキャッシュを消去してクレデンシャルを LDAP ディレクトリと照合します。クレデンシャルが有効である場合、Integration Server はそれをキャッシュします。有効でない場合、キャッシュは空のままです。 通常のセキュアな環境では、廃棄までの時間の値は 1 時間から 1 日が適当です。ハイセキュリティの環境では、廃棄までの時間を 1～5 分に設定します。 廃棄までの時間は絶対的な値であるので、アクティビティによってこれより長くクレデンシャルがキャッシュに残ることはありません。

7. [**設定内容の保存**] をクリックします。

   LDAP ディレクトリを使用するように Integration Server を設定したら、683 ページの「Integration Server に対する LDAP ディレクトリの定義 」の手順に進みます。

Integration Server に対する LDAP ディレクトリの定義

LDAP ディレクトリを Integration Server に定義するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**LDAP 設定**] をクリックします。
4. [**LDAP ディレクトリの追加**] をクリックします。

   注:
   LDAP ディレクトリを追加できるのは、プロバイダとして LDAP を選択した後だけです。

5. [**設定**] > [LDAP ディレクトリ] > [追加] ページで、次の情報を入力します。

パラメータ	指定する値
[ディレクトリの URL]	LDAP サーバの完全な URL。URL の形式は *protocol*://*hostname*:*portnumber* です。 ■ *protocol* は、標準接続の場合は LDAP、セキュア接続の場合は LDAPS です。 ■ *host* は、LDAP サーバのホスト名または IP アドレスです。*port* はサーバが稼動しているポートです。ポートは省略可能です。省略した場合、デフォルトポートとして LDAP の場合は 389、LDAPS の場合は 636 に設定されます。 たとえば、URL `ldaps://ldapserv1:700` を指定すると、ldapserv1 というホストの非標準ポート 700 で稼動する LDAP サーバへのセキュア接続が作成されます。 ldaps を指定した場合、Integration Server は SSL ソケットを使用してディレクトリサーバへのセキュア接続を作成しようとします。SSL を使用するように設定されているディレクトリサーバは、クライアントに対して自分自身を明らかにするサーバ認証を持っています。この認証は、信用のある機関によって署名されていないと有効と見なされません (すなわち、サーバ認証は CA によって署名されます)。デフォルトで、Integration Server は CA 認証が Integration Server の信用のある CA ディ

パラメータ	指定する値
	レクトリにある署名機関によって署名された認証のみを信用します。信用のある CA ディレクトリを設定する方法、および CA 認証を検索する方法については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
プリンシパル	Integration Server が LDAP サーバに接続する際に指定するユーザ ID。たとえば、`o=webm.com` または `dc=webm,dc=com` と指定します。 このユーザは、Administrator アカウントではなく、グループをクエリーする権限とグループメンバーシップを持つユーザとします。LDAP サーバが匿名アクセスを許可する場合は、このフィールドを空白のままにします。
[クレデンシャル]	Integration Server が LDAP サーバに接続する際に指定するパスワード。すなわち、プリンシパルのパスワード。Integration Server は、[**送信パスワード**] ページで指定された設定に従ってこのパスワードを暗号化します。詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
[接続タイムアウト (秒)]	Integration Server が LDAP サーバに接続するまで待機する秒単位の時間。この時間が経過すると、Integration Server はリストの次の設定済み LDAP サーバを試行します。デフォルトは 5 秒です。遅延の問題があるネットワークではこの数値を増やします。ほとんどの要求がバッチプロセスから来るのであれば、この数値を 30 秒以上に増やします。
[接続プールサイズの最小値]	Integration Server が LDAP サーバとの接続のために保持するプールで許可される接続の最小数。Integration Server が起動すると、最初にこの最小数の接続が接続プールに作成されます。Integration Server は、[**接続プールサイズの最大値**] フィールドに指定した最大許可数に達するまで、必要に応じて接続をプールに追加します。デフォルトは 0 です。
[接続プールサイズの最大値]	Integration Server が LDAP サーバとの接続のために保持するプールで許可される接続の最大数。Integration Server が起動すると最初に、[**接続プールサイズの最小値**] フィールドで指定した最小数の接続が接続プールに作成されます。Integration Server は、最大許可数に達するまで、必要に応じて接続をプールに追加します。デフォルトは 10 です。
[DN の合成]	ユーザ名にプリフィックスおよびサフィックスを追加することによって、識別名を作成します。 DN を合成する方法は、LDAP ディレクトリに対してクエリーを実行しないので、DN をクエリーする方法 (下記参照) より高速です。ただし、LDAP システムにすべてのユーザが単一フ

パラメータ	指定する値
	ラット構造で含まれているとは限らない場合は、DN をクエリーする方法を使用します。 **[DN プリフィックス]** LDAP サーバに渡す DN の先頭部分を指定するストリング。 **[DN サフィックス]** LDAP サーバに渡す DN の末尾部分を指定するストリング。 たとえば、プリフィックスが「cn=」、サフィックスが「,ou=Users」で、ユーザが「bob」を指定してログインすると、Integration Server は DN `cn=bob,ou=Users` を作成して認証のために LDAP サーバに送信します。 **注:** 適切な DN を作成するために必要なすべての文字を指定していることを確認してください。たとえば、上記の例のサフィックスでカンマを忘れた場合、すなわち、「`,ou=Users`」ではなく「`ou=Users`」と指定すると、Integration Server は無効な DN (`cn=bobou=Users`) を作成します。
[DN のクエリー]	ユーザの指定済みルートディレクトリを検索するクエリーを作成します。 LDAP ディレクトリが複合構造である場合は、DN を合成する方法 (上記参照) ではなく、この方法を使用します。 **[UID のプロパティ]** 「`cn`」、「`uid`」など、LDAP ユーザ ID を識別するプロパティ。 **[ユーザのルート DN]** 完全な識別名を入力します。たとえば、`ou=users,dc=webMethods,dc=com` を指定した場合、Integration Server は、ユーザがログインした名前と一致する共通名をルートディレクトリ `ou=users` で検索するクエリーを発行します。
[ユーザ電子メール]	ユーザの電子メールアドレスの保存に使用される属性名を指定します。たとえば、「mail」です。
[デフォルトのグループ]	ユーザが関連付けられる Integration Server グループ。ユーザには、この Integration Server グループのメンバーがアクセスできるサービスへのアクセスが許可されます。このアクセスは、グループが関連付けられている ACL によって制御されます。

パラメータ	指定する値
	[**グループメンバーの属性**] フィールドにも値を指定した場合、ユーザには、Integration Server グループのメンバーおよび Integration Server ACL にマッピングされている LDAP グループのメンバーと同じアクセス権が与えられます。 **重要:** グループ内に Administrator 特権を必要とするユーザがいる場合は、デフォルトグループに [**Anonymous**] を指定しないでください。デフォルトの ACL では Anonymous グループは拒否され、ルートページへのアクセスが許可されません。[**デフォルトのグループ**] フィールドで適切なグループを選択して、必要な ACL がグループに割り当てられるようにしてください。 **重要:** [**グループメンバーの属性**] フィールドまたは [**デフォルトのグループ**] フィールド、あるいはその両方のフィールドに値を指定する必要があります。
[グループメンバーの属性]	グループの各メンバーを識別するグループのディレクトリエントリの属性名。通常この値は「member」または「uniqueMember」ですが、LDAP ディレクトリのスキーマによって異なります。 Integration Server は、ログインしようとしているユーザが、ACL にマッピングされている LDAP グループに属しているのかを確認するために、ACL チェック時にこの情報を使用します。 ここに値を指定しない場合、Integration Server は LDAP グループのメンバーシップをチェックしません。そのため、ユーザが Integration Server サービスにアクセスできるかどうかは、[**デフォルトのグループ**] フィールドに指定されている Integration Server グループによって制御されます。 **注:** [**グループメンバーの属性**] フィールドまたは [**デフォルトのグループ**] フィールド、あるいはその両方のフィールドに値を指定する必要があります。
[グループ ID のプロパティ]	CN など、LDAP グループを識別するプロパティ。
[グループのルート DN]	完全な識別名。 たとえば、`ou=groups,webMethods,dc=com` と指定すると、Integration Server はすべての LDAP グループを表示するクエリーを発行します。

パラメータ	指定する値
	注: [**グループ ID のプロパティ**] および[**グループのルート DN**] フィールドに値を指定する必要があります。

6. [**変更内容の保存**] をクリックします。

   追加された LDAD ディレクトリが [LDAP ディレクトリの一覧] に表示されます。

7. [**上へ移動/下へ移動**] をクリックして、優先順位に基づいてリスト内のディレクトリの順序を変更します。

注:

- 複数の LDAP サーバを設定した場合、Integration Server による LDAP ディレクトリの検索は、[**セキュリティ**] > [ユーザ管理] > [LDAP 設定] ページに表示される順序で実行されます。Integration Server は、最初の LDAP ディレクトリでユーザを見つけられない場合、リストの順序で検索します。
- Integration Server および LDAP サーバ間の接続が断続的に中断され、トレースログで次の例外を見かける場合、標準 LDAP ポート (389) を使用するのではなく、LDAP サーバのグローバルカタログポート (3268/3269) に接続してください。例: `ldap://<hostname:3268>`

```
PartialResultException in the trace logs : [ISS.0002.0000T]
[LDAPv2] javax.naming.PartialResultException [Root exception is
javax.naming.CommunicationException:
[Root exception is java.net.SocketTimeoutException: connect timed out]]
```

- グローバルカタログポート (3268/3269) を使用しても接続の問題が続く場合、次のエラーが原因である可能性があります。
  - 接続タイムアウトエラー
  - 通信エラー
  - リソース不足エラー
  - グローバルカタログとして機能する孤立したドメイン

一時的なエラーの場合、接続を復元するため、`watt.server.ldap.retryCount` および `watt.server.ldap.retryWait` パラメータに適切な値を設定します。

LDAP ユーザのアクセス権の ACL へのマッピング

Integration Server グループと同様に、LDAP グループを ACL に関連付けることによって Integration Server リソースへのアクセスを制御することができます。LDAP グループを ACL に関連付けることを「マッピング」と呼びます。LDAP グループへの ACL のマッピングは、[**セキュリティ**] > [ACL] ページを使用して直接実行できます。ACL へのグループアクセスの許可の詳細については、539 ページの「ACL へのグループアクセスの許可または拒否」を参照してください。

外部ディレクトリとしての LDAP の使用の停止

外部ディレクトリとして LDAP を使用しなくなった場合は、設定を更新して、外部ディレクトリ設定を削除できます。

外部ディレクトリとしての LDAP の使用を停止するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**セキュリティ**] > [ユーザ管理] に移動します。
3. [**LDAP 設定の編集**] をクリックします。
4. [**プロバイダ**] フィールドの [**ローカル**] をクリックします。
5. [**設定内容の保存**] をクリックします。
6. [**OK**] をクリックします。
7. Integration Server を再起動します。

ユーザアカウントとグループに関する考慮事項

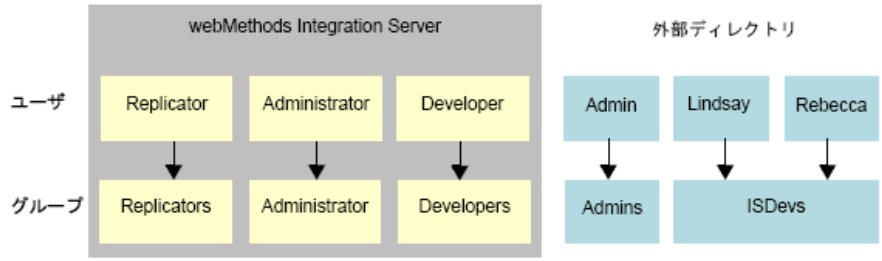
ここでは、ユーザ情報やグループ情報に外部ディレクトリを使用する場合に考慮すべきユーザアカウントとグループの情報について説明します。

- **内部と外部のユーザアカウントとグループ名を一意に保持する** 外部ユーザアカウントのユーザ名が内部ユーザアカウントのユーザ名と同一である場合や、外部グループのグループ名が内部グループのグループ名と同一である場合、混乱が生じる可能性があります。詳細については、689 ページの「内部と外部のユーザアカウントとグループ名を一意に保持する方法について」を参照してください。
- **Integration Server Administrator を使用して、セントラルユーザを管理 (作成、編集、削除) することはできない** セントラルユーザおよびディレクトリの管理には、My webMethods Server を使用する必要があります。詳細については、*Administering My webMethods Server* を参照してください。
- **Integration Server Administrator を使用して LDAP のユーザ情報およびグループ情報を管理 (作成、編集、削除) することはできない**LDAP ディレクトリの変更については、お使いのサイトのディレクトリ更新の一般的な手順に従ってください。
- **事前定義済みの Replicator アカウントを使用しなくてもパッケージの複製を実行できる** サブスクリプション要求者が Replicators ACL に割り当てられているグループのメンバーであるアカウントを指定している場合に限り、パッケージの複製に別のアカウントを使用できます。詳細については、689 ページの「ユーザグループおよびパッケージの複製について」を参照してください。

内部と外部のユーザアカウントとグループ名を一意に保持する方法について

Software AG では、内部のユーザカウントおよびグループと外部のユーザアカウントおよびグループとの間でユーザ名およびグループ名を一意に保持することをお勧めします。外部ユーザアカウントのユーザ名が内部ユーザアカウントのユーザ名と同一である場合や、外部グループのグループ名が内部グループのグループ名と同一である場合、混乱が生じる可能性があります。内部と外部のユーザ名またはグループ名がまったく同じである場合は、内部定義された情報が自動的に使用されます。

混乱を避けるために、Software AG では、外部ディレクトリを使用する際には、内部でユーザアカウントやグループをセットアップしないことをお勧めします。例外として、事前定義済みの Default、Administrator、Developer、Replicator の各ユーザアカウント、および事前定義済みの Everybody、Administrators、Developers、Replicators、Anonymous の各グループがあります。これらのユーザアカウントおよびグループは削除できません。事前定義済みの「内部」アカウントおよびグループの定義が正しいことを確認してください。



例外として、すべての内部定義ユーザは、内部定義された Everybody グループのメンバーになります。

ユーザグループおよびパッケージの複製について

Integration Server は事前定義済みの Replicator アカウントで配信されますが、パッケージの複製用に別のアカウントを使用することもできます。サブスクリプション要求者が Replicators ACL に割り当てられているグループのメンバーであるアカウントを指定している限り、そのユーザは複製を実行できます。

パッケージを別のサーバにパブリッシュする場合、パブリッシャーサーバは、サブスクリプション要求者によって指定されたアカウントを使用します。たとえば、パブリッシャーまたはサブスクライバのいずれかであるサブスクリプション要求者が DEPT01 というアカウントを指定した場合、パブリッシャーは、DEPT01 としてサブスクライバサーバにログインすることになります。DEPT01 は、サブスクライバサーバ上の Replicators ACL に割り当てられているグループのメンバーである必要があります。

パッケージの複製の詳細については、716 ページの「サーバ間でのパッケージのコピー」を参照してください。

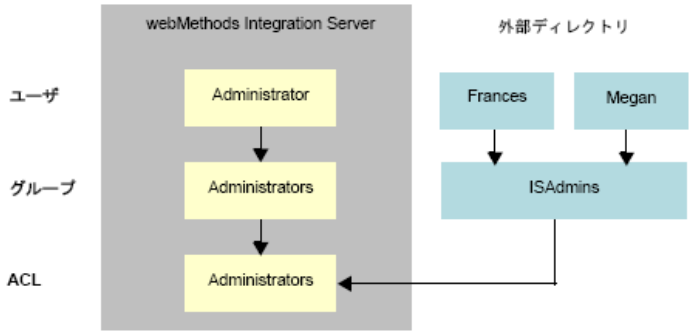
外部ユーザへの Administrator 特権の付与について

Administrators ACL は、Administrator 特権を制御します。外部定義されたユーザを内部定義されたグループに割り当てることはできないため、外部定義されたユーザに Administrator 特権を付与する場合に、その外部定義ユーザを内部定義の Administrators グループに割り当てるという方法は使用できません。外部定義されたユーザに Administrator 特権を付与するには、最初に、Administrator 特権用に外部定義のグループをセットアップする必要があります。次に、セットアップした外部定義の Administrators グループを Administrators ACL に追加します。

セントラルユーザのグループを Integration Server Administrators にするには、ユーザのグループまたは役割を以下の ACL に追加する必要があります。

- Administrators ACL
- Default ACL
- Developers ACL
- Internal ACL
- Replicators ACL
- Anonymous ACL (ユーザの役割/グループにまだこの ACL が割り当てられていない場合)

注:
セントラルユーザ管理を使用するように Integration Server を設定してある場合、Anonymous ACL には My webMethods ユーザの役割が自動的に組み込まれます。



外部定義ユーザへの Administrator 特権の付与

Administrator 特権を外部定義ユーザに付与するには

1. 特権を付与するユーザ用に外部定義のユーザアカウントがセットアップされていないときは、それをセットアップします。

2. 外部定義の Administrators グループがセットアップされていないときは、それをセットアップします。

重要:
外部定義グループの名前として「Administrators」は使用しないでください。外部定義グループの名前は、内部定義グループの名前と同じであってはなりません。

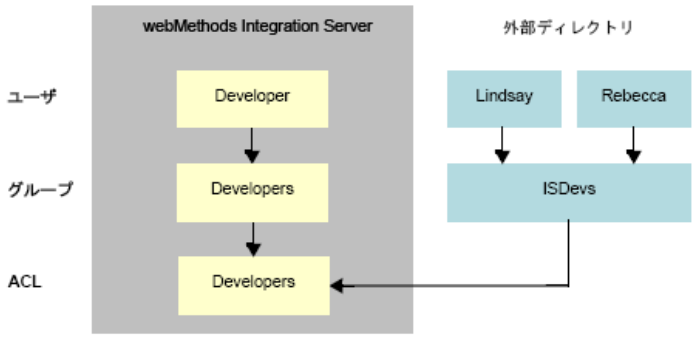
3. 外部定義のユーザを外部定義の Administrators グループ (上の図の ISAdmins) のメンバーに指定します。

4. 外部定義の Administrators グループが [**許可**] リストに含まれるように、Administrators ACL を更新します。

   外部定義の Administrators を [許可] リストに含める方法については、539 ページの「ACL へのグループアクセスの許可または拒否」を参照してください。

外部ユーザへの Developer 特権の付与

Developers ACL は Software AG Designer から Integration Server に接続してサーバ上にあるサービスを作成、変更および削除できるユーザを制御します。外部定義されたユーザを内部定義されたグループに割り当てることはできないため、外部定義されたユーザに Developer 特権を付与する場合に、その外部定義ユーザを内部定義の Developers グループに割り当てるという方法は使用できません。外部定義されたユーザに Developer 特権を付与するには、最初に、Designer 用に外部定義のグループをセットアップする必要があります。次に、セットアップした外部定義の Developers グループを Developers ACL に追加します。



Developer 特権を外部定義ユーザに付与するには

1. 特権を付与するユーザ用に外部定義のユーザアカウントがセットアップされていないときは、それをセットアップします。

2. 外部定義の Developers グループがセットアップされていないときは、それをセットアップします。

重要:
外部定義グループの名前として「Developers」は使用しないでください。外部定義グループの名前は、内部定義グループの名前と同じであってはなりません。

3. 外部定義のユーザを外部定義の Developers グループ (上の図の ISDevs) のメンバーに指定します。

4. 外部定義の Developers グループが [**許可**] リストに含まれるように、Developers ACL を更新します。

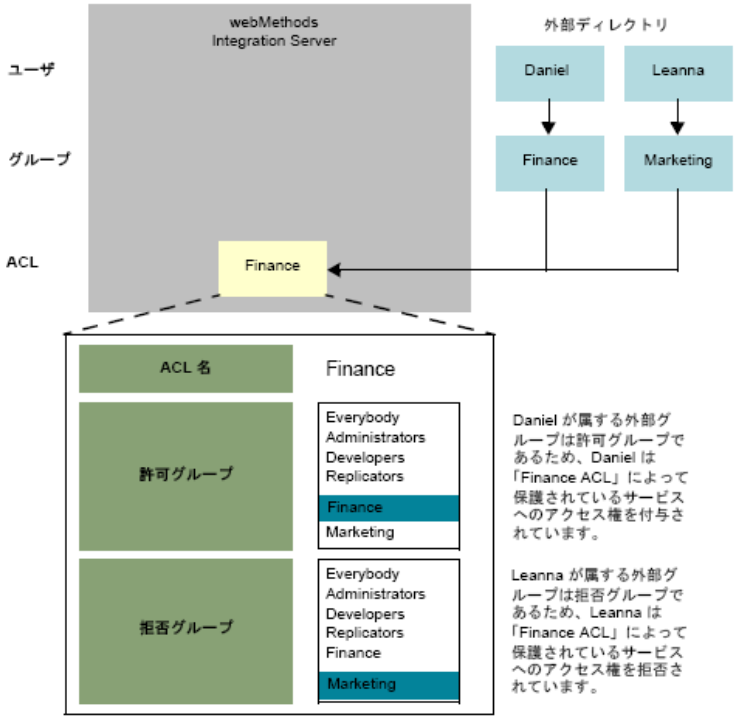
   外部定義の Developers を [許可] リストに含める方法については、539 ページの「ACL へのグループアクセスの許可または拒否」を参照してください。

外部ユーザへのサービスとファイルに対するアクセス特権の付与

サービスとファイルへのアクセスを制御する ACL を作成し、それを保護するサービスとファイルに割り当てます。

サーバがサービスやファイルへのアクセス権を付与する際には、最初に、内部定義された情報を使用して、そのクライアントが ACL のリストにある許可グループまたは拒否グループのいずれのメンバーであるかを判別します。内部定義された情報では見つからなかった場合、サーバは外部定義された情報を取得して、そのクライアントの ACL がアクセスを許可しているか拒否しているかを判別します。

外部定義されたユーザに対してサービスやファイルへのアクセス権を付与するには、そのサービスやファイルを保護している ACL の許可グループに目的の外部ユーザのグループまたは役割を含めます。同様に、外部定義されたユーザに対してサービスやファイルへのアクセス権を明示的に拒否するには、そのサービスやファイルを保護している ACL の拒否グループに目的の外部ユーザのグループまたは役割を含めます。


webMethods
Integration Server
外部ディレクトリ
ユーザ
グループ
ACL
Daniel
Leanna
Finance
Marketing
Finance
ACL 名
Finance
許可グループ
Everybody
Administrators
Developers
Replicators
Finance
Marketing
拒否グループ
Everybody
Administrators
Developers
Replicators
Finance
Marketing
Daniel が属する外部グループは許可グループであるため、Daniel は「Finance ACL」によって保護されているサービスへのアクセス権を付与されています。
Leanna が属する外部グループは拒否グループであるため、Leanna は「Finance ACL」によって保護されているサービスへのアクセス権を拒否されています。

37 パッケージの管理

- パッケージの使用 696
- サーバによるパッケージ情報の保存 700
- パッケージに関する情報の検索 703
- パッケージの使用 711
- サーバ間でのパッケージのコピー 716
- パッケージクラスローダの使用 742
- パッケージのホット展開 742
- パッケージの自動展開 746

パッケージの使用

パッケージには、サービスおよび関連ファイルのセット (仕様、ドキュメントタイプ、DSP など) が含まれています。サービス、仕様、ドキュメントタイプまたは DSP を webMethods Integration Server に追加する場合は、それを必ずパッケージに追加してください。パッケージは、サービスと関連ファイルをグループ化するために使用します。

複数の関連ファイルを 1 つのパッケージとしてまとめれば、そのパッケージ内のすべてのサービスとファイルを 1 つのユニットとして簡単に管理することができます。たとえば、1 回の操作でそれらをすべて有効/無効にしたり、リフレッシュしたり、削除したりすることができます。また、複数の Integration Server をインストールしている場合は、パッケージ管理機能を使用してパッケージ内の一部またはすべてのサービスやファイルを別のサーバにコピーすることができます。

ユーザがサービスをグループ化する際に使用するパッケージ構造の種類は任意ですが、サービスを機能別または用途別に分けてパッケージを作成するのが普通です。たとえば、購買関連サービスはすべて「PurchaseOrderMgt」という名前のパッケージに、時間報告サービスはすべて「TimeCards」という名前のパッケージにそれぞれ入れることができます。

重要:
サーバ上のすべてのサービスはパッケージに属している必要があります。サービスを実行できるようにするには、その所属先のパッケージをロードする必要があります。

パッケージとその内容へのアクセスは、ACL で制御されます。ACL では、Integration Server Administrator および Designer からのパッケージの表示、パッケージ内容の編集またはパッケージ内のサービスの実行に関する権限を制御します。パッケージ保護の詳細については、534 ページの「ACL によるリソースへのアクセス制御」を参照してください。

パッケージを特定のポートに関連付けることにより、パッケージを複製しても同じ番号のポートを新しいサーバで引き続き使用できます。ポートとパッケージの関連付けの詳細については、169 ページの「ポートについて」を参照してください。

事前定義済みのパッケージ

Integration Server インスタンスごとに、次の事前定義済みパッケージがデフォルトでインストールされます。これらは、パッケージリポジトリにも含まれています。

パッケージ	説明
デフォルト	ユーザがパッケージ名を指定せずに (たとえば、http://localhost:5555 という URL を使用して) サーバにアクセスした場合、Integration Server はこのパッケージにアクセスします。また、最初にパッケージを作成せずに、作成するエレメントを保存するために使用することもできます。 **注:** Integration Server は、Default パッケージの pub ディレクトリで index.dsp ファイルまたは index.html ファイルを検索します。出荷時には、pub ディレクトリに index.html ファイルが含まれており、ユーザはこのファイルを使用して WmRoot パッケージの index.dsp ファ

パッケージ	説明
	イルを参照することができます。index.dsp ファイルは Integration Server Administrator をロードします。ユーザが、誤って Integration Server Administrator にアクセスするのを防止するため、Default/pub の index.html ファイルを編集し、無難なページを参照するように変更することができます。ユーザが、Integration Server 上のすべての DSP ファイルを表示したりファイルにアクセスしたりするのを防止するため、watt.server.displayDirectories サーバ設定パラメータを使用できます。詳細については、1091 ページの「ステージ 7: セキュリティの設定」を参照してください。
WmPublic	このパッケージには、ユーザがクライアントアプリケーションおよびサービスから呼び出すことができるサービスが収められています。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。
WmRoot	このパッケージは、Integration Server のコア機能および補助ファイルを提供します。 **重要:** このパッケージは、変更または削除しないでください。
WmAdmin	このパッケージは Integration Server Administrator および Integration Server Administrator API の機能とファイルを提供します。Administrator API の詳細については、1029 ページの「Integration Server Administrator API」を参照してください。 **重要:** このパッケージは、変更または削除しないでください。
WmART	このパッケージは webMethods 6 以上のアダプタをサポートします。 **重要:** WmArt パッケージはデフォルトでインストールされますが、このパッケージを使用する完全なライセンスがない場合、パッケージは非表示になり、カスタムアダプタを作成または実行したり、トランザクションサービスを使用したりすることはできません。
WmXSLT	このパッケージには、XML データの形式または構造を変換するためのサービスが収められています。詳細については、『*webMethods Integration Server Built-In Services Reference*』および『*webMethods Service Development Help*』を参照してください。
WmJSONAPI	このパッケージには、JSON API に準拠するために REST V2 リソースをサポートするサービスが含まれています。
WmParquet	このパッケージには、Integration Server での parquet ファイルの処理をサポートするサービスが含まれています。詳細については、

パッケージ	説明
	『*webMethods Integration Server Built-In Services Reference*』を参照してください。

次の表に示すパッケージは、ユーザまたは他の webMethods 製品が特定のタスクを実行するためのサービスを提供します。

パッケージ	パッケージに含まれるサービスの用途	参照先
WmARTExtDC WmISExtDC WmTNExtDC	これらのパッケージには、Infrastructure Data Collector が、Integration Server にインストールされているアダプタの検出と監視、Integration Server 自体と Trading Networks Server の監視を行うためのサービスが収められています。	*Administering webMethods Optimize*
WmAssetPublisher	このパッケージには、Integration Server がサービスに関するメタデータの抽出と CentraSite Metadata に対するパブリッシングを行うために使用するサービスが収められています。	*webMethods BPM and CAF Workspace Metadata Help*
WmCloud	このパッケージには、Integration Server が webMethods Integration Cloud とサービスを共有するために使用するサービスが収められています。	*Configuring On-Premise Integration Servers for webMethods Cloud*
WmDesigner	このパッケージには、Software AG Designer でモデル化されているビジネスプロセスをサポートするサービスが収められています。	*Software AG Designer Online Help*
WmFlatFile	このパッケージには、フローサービスまたはファイルポーリング処理サービスが、受信したフラットファイルを受け入れおよび使用するために最初に呼び出すサービスが収められています。	*webMethods Integration Server Built-In Services Reference*
WmGRPC	このパッケージには、Integration Server での gRPC をサポートするサービスが収められています。	および『*webMethods Service Development Help*』
WmMobileSupport	このパッケージには、モバイルデバイスとバックエンドアプリケーションとの間のデータ同期ソリューションをサポートするサービスが含まれています。 **注:**	*Developing Data Synchronization Solutions with webMethods Mobile Support*

パッケージ	パッケージに含まれるサービスの用途	参照先
	WmMobileSupport パッケージおよび webMethods Mobile Support 機能は廃止されました。	
WmOptimize	このパッケージには、webMethods Optimize を使用して監視されているビジネスプロセスをサポートするサービスが収められています。	Optimize のマニュアル
WmPRT	このパッケージには、webMethods Process Engine を使用して実行されるビジネスプロセスをサポートするサービスが収められています。	*Administering webMethods Process Engine*
WmTaskClient	このパッケージには、webMethods Task Engine を使用して開発されたタスクをサポートするサービスが収められています。	*webMethods Task Engine User's Guide*
WmWin32	Integration Server で統合 Windows 認証を使用できます。 **注:** このパッケージは、Integration Server 7.1 で廃止されました。	1293 ページの「Integration Server がクライアントとして機能するときの NTLM 認証の使用」

Integration Server を効率化する場合は (アダプタをホストするだけに利用するなど)、上記のパッケージの多くを無効にできます。

重要:
Default、WmAdmin、WmPublic、WmRoot のパッケージは無効化、削除または変更しないでください。

次のパッケージは無効にできます。

パッケージ	制限
WmAdmin	**重要:** Integration Server Administrator API を使用する場合、WmAdmin パッケージを無効にしないでください。
WmART	**重要:** WmARTExtDC を無効にしない限り、このパッケージは無効にしないでください。無効にすると、WmARTExtDC がロードされません。
WmAssetPublisher	なし。

パッケージ	制限
WmARTExtDC WmISExtDC WmTNExtDC	**重要:** WmTNExtDC および WmARTExtDC を無効にしない限り、WmISExtDC パッケージは無効にしないでください。無効にすると、これらのパッケージがロードされません。
WmFlatFile	なし。
WmGRPC	組み込み gRPC サーバの実行、gRPC クライアント要求の処理、または gRPC 記述子の作成が必要な場合、WmGRPC パッケージを無効にしないでください。
WmPRT WmDesigner WmTaskClient	なし。
WmMobileSupport	WmMobileSupport パッケージを無効にする前に、有効になっているすべてのモバイル同期コンポーネントを一時停止してください。
WmOptimize	なし。
WmXSLT	なし。
WmJSONAPI	なし。
WmParquet	なし。

パッケージの無効化については、713 ページの「パッケージの無効化」を参照してください。

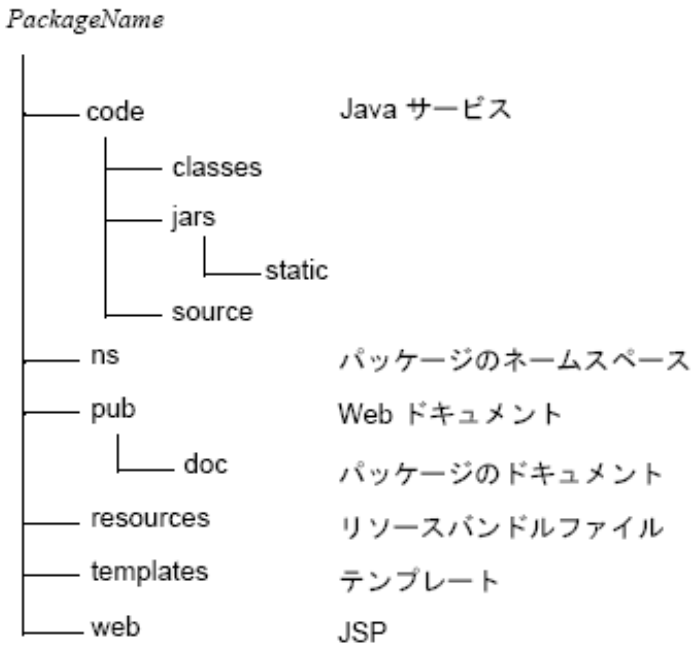
パッケージリポジトリ

Software AG_directory \IntegrationServer\packages ディレクトリは、任意の Integration Server インスタンスにインストールできるパッケージのリポジトリとして機能します。リポジトリには、事前定義済みのパッケージと、特定のタスクを実行するために使用できる、その他多くのパッケージが含まれています。パッケージリポジトリからパッケージをインストールまたは更新する方法については、71 ページの「サーバインスタンスでのパッケージの更新」を参照してください。

サーバによるパッケージ情報の保存

サーバは、パッケージ情報を *Integration Server_directory* \instances*instance_name*\packages ディレクトリに物理的に保存します。また、パッケージごとに新しいサブディレクトリを作成します。サブディレクトリの名前はパッケージ名と同じです。たとえば、パッケージ名が「TimeCards」であれば、 *Integration Server_directory* \instances*instance_name*\packages\TimeCards ディレクトリが作成されて、パッケージのファイルがそこに保管されます。

ユーザが新しいパッケージを作成すると以下のサブディレクトリが作成され、パッケージに関するファイルがすべてこのサブディレクトリに保管されます。



- **code サブディレクトリ** このパッケージに属する Java サービスおよび C/C++ サービスが保存されます。code サブディレクトリ内には、下記の classes、jars、static、source および lib サブディレクトリがあります。
  - **classes サブディレクトリ** Java サービスおよび C/C++ サービスの Java クラスが保存されます。
  - **jars サブディレクトリ** jar ファイル内に一緒にパッケージされている Java クラスが保存されます。
  - **static サブディレクトリ** jar ファイル内に一緒にパッケージされている Java クラスが保存されます。jar ファイルを Integration Server の他のパッケージで使用可能にする場合、さらに、他の Integration Server システムのパッケージでも使用可能にする場合に、jar ファイルをこの場所に配置します。

    jar ファイルを static サブディレクトリに配置すると、起動時に Integration Server はこれらのファイルをサーバクラスパスに自動的にロードします。また、直近のパッケージが無効であっても、jar ファイルは他のパッケージで使用できます。

    注:
    新規パッケージを作成しても、Integration Server は static サブディレクトリを自動的には作成しません。このサブディレクトリはユーザ定義です。
  - **source サブディレクトリ** Java サービスのソースが保存されます。
  - **libs サブディレクトリ** Java サービスや C/C++ サービスで使用する DLL ライブラリまたは特殊ライブラリが保存されます (上の図には表示されていません)。

    注:
    libs ディレクトリが存在すると、パッケージを再ロードする際にサーバの再起動が必要になるので、Integration Server は libs ディレクトリを自動的には作成しません。共

有ライブラリを使用するパッケージを再ロードするには、サーバの再起動が必要になります。

管理を円滑に行うために、共有ライブラリを使用するサービスは同じパッケージに入れます。

- **config サブディレクトリ** 次のファイルが保存されます。
  - **listeners.cnf**このファイルには、パッケージに対して定義したポートや、各ポートを介してアクセスできるサービスに関する情報が含まれています。ポートの設定およびポート経由でのサービスへのアクセス許可の詳細については、167 ページの「ポートの設定」を参照してください。
  - **urlalias.cnf**このファイルには、パッケージのサービスに関連付けられている HTTP URL エイリアスの定義が含まれています。エイリアスがない場合、このファイルは存在しません。urlalias.cnf ファイルは手動で更新しないでください。HTTP URL エイリアスの詳細については、449 ページの「HTTP URL エイリアスの設定」を参照してください。
- **ns サブディレクトリ** フローサービス、仕様、ドキュメントタイプ、スキーマ、トリガー、アダプタ通知、アダプタドキュメント、アダプタサービス、アダプタコネクタおよび Java サービスのコードフラグメントが保存されます。
- **pub サブディレクトリ** パッケージに関する Web ドキュメントが保存されます。パッケージの Web ドキュメントにアクセスする方法については、710 ページの「パッケージに関するドキュメントの表示」を参照してください。
- **doc サブディレクトリ** パッケージに関するドキュメントが保存されます。
- **resources サブディレクトリ** ユーザデータではないアプリケーションデータなどのリソースバンドル *<bundle.properties>* が保存され、Integration Server アプリケーションとは別にされます。次のアイテムがバンドル内の典型的なリソースです。
  - アイコン
  - ウィンドウの位置
  - ダイアログボックスの定義
  - プログラムテキスト
  - メニュー

  アプリケーション全体を再インストールすることなく、Integration Server の外観をさまざまに変更および更新することが簡単にできます。Integration Server の日本語 Language Pack は、リソースバンドルの一例であり、日本語バージョンのサーバ用の言語ファイルおよびイメージファイルを含みます。
- **templates サブディレクトリ** このパッケージに関連する出力テンプレートが保存されます。
- **web サブディレクトリ** このパッケージに関連する JSP テンプレートが保存されます。

マニフェストファイル

各パッケージにはマニフェストファイルが収められています。これには次のものが含まれます。

- **パッケージが有効か無効かの表示**サーバの初期化の際、無効なパッケージはロードされません。また、無効なパッケージ内にあるエレメントにはアクセスできません。
- **パッケージ内にある開始サービス、シャットダウンサービス、複製サービスのリスト**開始サービス、シャットダウンサービス、複製サービス、およびその識別方法については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
- **パッケージの説明**パッケージの簡単な説明。
- **バージョン情報**パッケージのバージョンとビルド番号。パッケージがパブリッシュされた JVM バージョンも含まれます。
- **当てられたパッチ**パッケージに当てられたパッチのリスト。インストールに関連した名称または番号が付いています。このパッチ名またはパッチ番号は、障害追跡システムで使用されているものを適用していることがよくあります。
- **パッケージの依存性**開発者は、特定パッケージ内のエレメントに先だってロードしておく必要のある他のパッケージを指定できます。つまり、開発者はパッケージ間の依存関係を指定できます。詳細については、707 ページの「パッケージに関する情報の表示」および『*webMethods Service Development Help*』を参照してください。
- **ターゲットパッケージ名**パッケージの名前。
- **パブリッシャーサーバ**パッケージをパブリッシュした Integration Server。パッケージがパブリッシュされていないと、このフィールドには「なし」と記載されます。

パッケージのマニフェストは、パッケージ用のトップディレクトリの manifest.v3 ファイル内にあります。

パッケージに関する情報の検索

サーバは、パッケージに関するさまざまな情報を表示します。ここでは、使用可能な情報とその情報を表示するための手順を示します。

情報	参照ページ
サーバ上にある全パッケージのリスト	704 ページの「サーバ上にあるパッケージの表示」
サーバ上の指定されたパッケージのリスト	704 ページの「サーバ上にあるパッケージの表示」
パッケージが正常にロードされたかどうかの状態	706 ページの「パッケージが正常にロードされたかどうかの判定」

情報	参照ページ
パッケージの有効性/無効性に関する状態	707 ページの「パッケージが有効か無効かの判定」
パッケージのバージョン番号	707 ページの「パッケージに関する情報の表示」
メモリに正常にロードされたパッケージ内のエレメントの数	707 ページの「パッケージに関する情報の表示」
パッケージリストを表示できるユーザを指定するための ACL の名前	707 ページの「パッケージに関する情報の表示」
メモリにロードできなかったパッケージ内のエレメントのリスト	707 ページの「パッケージに関する情報の表示」
パッケージに関するロードエラーのリスト	707 ページの「パッケージに関する情報の表示」
パッケージ内にある開始サービス、シャットダウンサービス、複製サービスのリスト	707 ページの「パッケージに関する情報の表示」
パッケージの依存先パッケージのリスト	707 ページの「パッケージに関する情報の表示」
当該パッケージにサブスクライブするサーバのリスト	707 ページの「パッケージに関する情報の表示」
パッケージに関するマニュアル	710 ページの「パッケージに関するドキュメントの表示」

パッケージに関連付けられた HTTP URL エイリアスのリストを表示するには、[**設定**] > [URL エイリアス] ページを使用します。HTTP URL エイリアスの詳細については、449 ページの「HTTP URL エイリアスの設定」を参照してください。

サーバ上にあるパッケージの表示

Integration Server Administrator のメインパッケージ管理ページには、サーバ上にある全パッケージのリストが表示されます。また、パッケージが正常にロードされたかどうか、およびパッケージが有効かどうかも表示されます。

注:
サーバには、ユーザがリストへのアクセスを許可されているパッケージのみが表示されます。

サーバ上にあるパッケージを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**パッケージ**] > [管理] に移動します。

パッケージリストのフィルタ処理

メインパッケージ管理ページには、デフォルトでサーバ上にある全パッケージのリストが表示されます。フィルタを使用し、表示されるパッケージを制限することで、管理しやすい簡潔なリストを作成できます。次の方法を使用して、[**パッケージの一覧**] を管理できます。

- パッケージ名全体または名前の一部を指定し、指定された条件に一致するパッケージを含めるか除外することで、パッケージのデフォルトリストをフィルタ処理します。
- その結果を再度フィルタ処理し、リストをさらに絞り込みます。

> **ヒント:**
> フィルタ処理を無効にし、サーバ上の全パッケージを示すデフォルトリストに戻るには、[**すべてのパッケージを表示する**] をクリックします。

パッケージリストをフィルタ処理するには

1. [**パッケージ**] > [管理] に移動します。
2. [**パッケージをフィルタする**] をクリックします。[**パッケージの一覧**] の上部にフィルタ処理オプションが表示されます。

> **注:**
> [**パッケージをフィルタする**] が有効化されている間は、Integration Server に対するすべての変更 (新規パッケージなど) は [**パッケージの一覧**] に反映されません。[**すべてのパッケージを表示する**] をクリックして通常モードに戻ると、リストが更新されます。

3. 以下のオプションの一部またはすべてを選択します。

オプション	説明
[フィルタ条件]	フィルタに対して指定するストリング。デフォルトでは、ストリングと一致した名前のパッケージが結果に含められます。フィルタ条件には、リテラルや、リテラルとワイルドカード文字の組み合わせを使用できます。ワイルドカード文字としてサポートされているのは、「*」 (アスタリスク) と「?」 (疑問符) だけです。フィルタ条件を空白にした場合は、すべてのパッケージが結果に含められます。 **重要:** [**フィルタ条件**] フィールドに入力されたパッケージ名の大文字と小文字は区別されます。たとえば、「wma*」と入力した場合、「WmA」で開始するパッケージはすべて無視されます。
[有効なパッケージを含む] **[無効なパッケージを含む]**	有効化されているパッケージ ([**パッケージの一覧**] の [**有効**] 列に [**はい**] と指定されているパッケージ) のみを含めるか、無効化されているパッケージ ([パッケージの一覧] の [**有効**] 列に [**いいえ**] と指定されているパッケージ) のみを含めるか、または有効化されているパッ

オプション	説明
[両方含む]	ケージと無効化されているパッケージの両方を含めるかを指定します。
[結果をフィルタする]	既にフィルタ処理されたリストがある場合に、デフォルトリストではなく、この結果リストに再度フィルタをかける場合は、このオプションを有効にします。
[結果から除外する]	[**フィルタ条件**]に一致するパッケージではなく、一致しないパッケージを表示する場合は、このオプションを有効にします。

4. [**サブミット**] をクリックします。フィルタオプションに一致するパッケージのみが表示されます。

フィルタ処理済みパッケージリストの絞り込み

フィルタ処理済みのパッケージリストからパッケージをフィルタ処理するには

1. 705ページの「パッケージリストのフィルタ処理」の説明に従って [**パッケージの一覧**] をフィルタ処理します。フィルタ条件に一致するパッケージが表示されます。

2. [**結果をフィルタする**] モードを有効化します。こうすることで、サーバ上の全パッケージのデフォルトリストではなく、現在表示されているパッケージリストのみを対象に検索が実行されます。

   注:
   [**結果から除外する**] オプションを有効化して、フィルタ条件に一致するパッケージではなく、一致しないパッケージを表示することもできます。

3. 新しいフィルタ条件を入力し、[**サブミット**] をクリックします。[**結果をフィルタする**] モードを毎回有効化することを忘れずに、必要なだけ何度でも操作を繰り返します。

パッケージが正常にロードされたかどうかの判定

[**パッケージ**] ページの [**ロード済み**] 列に状態が表示されます。

状態	説明
✓ はい	サーバは、パッケージに関連するすべてのエレメントを正常にロードしました。パッケージ内のエレメントは使用可能です。サーバは、パッケージが空の場合もこの状態を表示します。
✓ 部分的	パッケージに関するエレメントのうち、ロードされなかったものがありました。正常にロードされたエレメントは使用可能です。正常にロードされなかったエレメントとその理由を表示する方法については、707ページの「パッケージに関する情報の表示」を参照してください。

状態	説明
いいえ	サーバは、パッケージに関連するエレメントを 1 つもロードしませんでした。エレメントはいずれも使用できません。サーバがエレメントをロードできなかった理由を表示する方法については、707 ページの「パッケージに関する情報の表示」を参照してください。

サーバを起動すると、有効なパッケージに含まれるエレメントはすべて自動的にメモリにロードされます。サーバ起動時にパッケージが無効の場合は、パッケージが有効になった時点でエレメントがロードされます。必要に応じて、手動でパッケージを再ロードすることもできます。パッケージの再ロードについては、713 ページの「パッケージの再ロード」を参照してください。

パッケージが有効か無効かの判定

[**パッケージ**] ページの [**有効**] 列に状態が表示されます。ここでは、パッケージが有効か無効かが表示されます。クライアントは、パッケージが有効になって初めてそのパッケージ内のサービスにアクセスできます。

状態	説明
✓ はい	パッケージは有効なので、クライアントはそのパッケージ内のエレメントにアクセスすることができます。
いいえ	パッケージは無効なので、クライアントはそのパッケージ内のエレメントにアクセスすることはできません。
警告	パッケージ内の一部のエレメントに警告が発生しましたが、既にロードされているので使用できます。どのエレメントで警告が発生したかを確認する場合は、ページ下部の [ロード警告] リストを調べます。

パッケージの有効化および無効化の説明については、713 ページの「パッケージの有効化」および 713 ページの「パッケージの無効化」を参照してください。

パッケージに関する情報の表示

Integration Server にインストールされている任意のパッケージに関する情報を表示できます。

パッケージに関する情報を表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. ページの [**パッケージの一覧**] 領域で、情報を表示するパッケージの名前をクリックします。

4. [**パッケージ**] > [管理] > [*PackageName*] ページが表示されます。これらのフィールドの詳細については、708 ページの「パッケージ情報」を参照してください。

パッケージ情報

Integration Server は、インストールされた各パッケージに関する次の情報を保持します。

フィールド	説明
[パッケージ名]	パッケージの名前。
[バージョン]	パッケージのバージョン番号。
[ビルド]	パッケージを生成するたびに開発者がそのパッケージに割り当てる番号。たとえば、バージョン 1.0 のオーダパッケージを 10 回生成する場合に、開発者はその都度ビルド番号を 1、2、3...10 と割り当てます。ビルド番号は、通常、開発環境でパッケージの生成を特定するために使用されます。
[JVM バージョンの最小値]	このパッケージを実行するために必要な JVM (Java Virtual Machine : Java 仮想マシン) の最小バージョン。 パッケージのインストール先の Integration Server は、パッケージで必要な JVM バージョンと同じかそれ以降のメジャーバージョンの JVM で実行されている必要があります。たとえば、パッケージリリースの指定 JVM バージョンが 1.8.0_32 である場合、インストール先 Integration Server は 1.8 以降の JVM で実行されている必要があります。Integration Server JVM のメジャーバージョンがパッケージで必要な JVM より以前のバージョンである場合、Integration Server はパッケージをインストールしますが、アクティブにしません。
[パッケージリスト ACL]	パッケージに割り当てられるアクセスコントロールリスト。この ACL に関連付けられたユーザは、Integration Server Administrator または Designer にリストアップされたパッケージを見ることができます。パッケージ内のフォルダやエレメントを見るには、そのフォルダやエレメントへのリストアクセス権が必要です。
[含まれているパッチ]	このリリースのパッケージに適用されたパッチのリスト。これらはインストールにとって重要な番号であり、障害追跡システムから取得できます。
[説明]	パッケージの説明とその用途。
[パブリッシャー]	パッケージをパブリッシュした会社、組織またはサーバの名前。 **注:** デフォルトでは、パッケージリリースを作成した場合に限り、Integration Server がこのフィールドにパブリッシャーサーバ名を自動的に入力します。
[作成日時]	パッケージが作成された日付、時刻および年。

フィールド	説明
	注: デフォルトでは、パッケージリリースを作成した場合に限り、Integration Server がこのフィールドに日付、時刻および年を自動的に入力します。
[ロードされたエレメント]	サーバが正常にロードしたエレメントの数。サーバが正常にロードしたエレメントを表示するには、*<PackageName>* リンクの [**サービスを表示**] をクリックします。
[ロードされなかったエレメント]	サーバがロードできなかったエレメントの数。ロードされなかったエレメントが少なくとも 1 つあると、そのエレメントとロードされなかった理由がページ上の [**ロードエラー**] セクションに表示されます。
[開始サービス]	管理者が開始サービスとして識別したサービスのリスト。開始サービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
[シャットダウンサービス]	管理者がシャットダウンサービスとして識別したサービスのリスト。シャットダウンサービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
[複製サービス]	管理者が複製サービスとして識別したサービスのリスト。複製サービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
[このパッケージが依存しているパッケージ]	このパッケージより前にロードする必要があるパッケージのリスト。パッケージの依存性の詳細については、『*webMethods Service Development Help*』を参照してください。
[このパッケージに依存しているパッケージ]	このパッケージに依存するパッケージのリスト。このパッケージを無効にすると、その依存パッケージが影響を受けます。
[サブスクライバ]	このパッケージをサブスクライブする他の Integration Server のリスト。サーバ間でパッケージをコピーする方法、パッケージをサブスクライブする方法およびパッケージを別のサーバにパブリッシュする方法については、716 ページの「サーバ間でのパッケージのコピー」を参照してください。
[ロードエラー]	パッケージのインストール時にエラーが発生してサーバにロードできなかったエレメントのリストを表示します。一部のエレメントがロードされないと、パッケージのロード状態は [**部分的**] となります。
[ロード警告]	パッケージのインストール時に警告が発生したエレメントのリストを表示します。サーバは、警告状態が発生してもパッケージをロードしています。警告時にパッケージエレメントがロードされると、そのパッケージのロード状態は [**警告**] となります。

フィールド	説明
[パッチの履歴]	このリリースのパッケージに適用されたパッチまたは部分パッケージのリスト。

パッケージ内のサービスおよびフォルダに関する情報の表示

パッケージ内のサービスまたはフォルダのリストは参照することができます。752ページの「サービスとフォルダに関する情報の検索」を参照してください。

パッケージに関するドキュメントの表示

Integration Serverから提供されるWebドキュメントを使用して、パッケージの機能やエレメントに関するドキュメントを作成できます。Webドキュメントは、パッケージ用のpubサブディレクトリに格納されます。

パッケージ用のホームページと、パッケージの他のWebドキュメントへのリンクを含むindex.htmlファイルは必ず作成してください。

パッケージ用のホームページにアクセスするには

1. Integration Server Administratorを開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. パッケージのホームアイコン をクリックします。

パッケージのWebドキュメントへのアクセス

パッケージ用のWebドキュメントにアクセスするには

1. パッケージが有効であることを確認します(707ページの「パッケージが有効か無効かの判定」を参照してください)。
2. Webドキュメントの URLを入力します。Webドキュメントの URLは次の形式をとります。

   http://*host:port*/*PackageName*/*Docname*

   ここで、

host:port	Integration Server のサーバ名とポートアドレス。
PackageName	Webドキュメントが保存されているパッケージの名前。パッケージ名を指定しない場合、サーバはまず Default パッケージの Pub ディレクトリにアクセスします。

DocName	Web ドキュメントの名前。ドキュメント名を指定しない場合は、指定されたパッケージの Pub ディレクトリにある index.dsp ファイルまたは index.html ファイルが表示されます。

パッケージの使用

次のタスクでは、パッケージ内のすべてのファイルを 1 つの単位として処理することができます。

タスク	目的	参照先
作成	新しいパッケージを作成する。開発者は、Designer を使用してパッケージを作成します。詳細については、*webMethods Service Development Help*を参照してください。	712 ページの「パッケージの作成」
アクティブ化	サーバを再起動せずに、Server/packages ディレクトリに手動で移動したパッケージを使用する。	712 ページの「パッケージのアクティブ化」
再ロード	サーバを再起動せずにパッケージ内のサービスをメモリに再ロードする。	713 ページの「パッケージの再ロード」
有効	無効になっているパッケージを有効にする。	713 ページの「パッケージの有効化」
無効	パッケージを削除せずに、パッケージへのアクセスを無効にする。	713 ページの「パッケージの無効化」
削除	パッケージ内のすべてのサービスと関連ファイルをすべて削除する。	714 ページの「パッケージの削除」
回復	削除したパッケージからサービスと関連ファイルを回復する。	715 ページの「パッケージの回復」
アーカイブ	パッケージの作業コピーを作成する。通常は、リリースによって他の人がこのコピーを使用することはできません。このコピーはバックアップとして使用できます。	716 ページの「パッケージのアーカイブ」
コピー	サーバ間でパッケージをコピーする。	716 ページの「パッケージのアーカイブ」

注:
組み込みサービスのセットを使用してパッケージを管理することもできます。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

パッケージの作成

開発者は、サービスと関連ファイルの新しいグループを作成するときに 1 つのパッケージを作成します。これにより、サービスを保存できる空のコンテナと、関連ファイルが作成されます。パッケージが作成されると、700 ページの「サーバによるパッケージ情報の保存」でも説明したように、サーバはパッケージのディレクトリ構造を構築します。パッケージを作成する手順については、『*webMethods Service Development Help*』を参照してください。

パッケージのアクティブ化

パッケージは、Integration Server にインストールされている場合でもアクティブになっていない場合があります。パッケージはアクティブになって初めて Integration Server に公式に認められ、[**パッケージの管理**] ページの [**パッケージの一覧**] に表示されます。非アクティブなパッケージは packages ディレクトリ内に存在しますが、サーバに公式には認められていません。

パッケージが非アクティブになるのは、次の理由が考えられます。

- Integration Server の実行中にパッケージを手動でインストールした。
- 使用サーバに別のサーバがパッケージをパブリッシュしたが、そのパッケージは使用サーバ上の JVM バージョンより高いバージョンを必要としている。サブスクライバサーバは、こういった状況下ではパッケージをアクティブ化しません。

  サブスクライブする Integration Server は、パッケージで必要な JVM バージョンと同じかそれ以降のメジャーバージョンの JVM で実行されている必要があります。たとえば、パッケージリリースの指定 JVM バージョンが 1.8.0_32 である場合、インストール先 Integration Server は 1.8 以降の JVM で実行されている必要があります。
- インストールしたパッケージが、サーバに存在しないか、または無効にされている別のパッケージに対する依存性を持つ。パッケージが無効にされている場合、サーバはパッケージをインストールしますが、アクティブにはしません。依存性が満たされると、パッケージをアクティブにできます。

パッケージは、サーバの再起動またはパッケージのアクティブ化を行わない限り使用できません。

パッケージをアクティブにするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**非アクティブのパッケージをアクティブにする**] をクリックします。
4. [**非アクティブのパッケージ**] 領域のプルダウンメニューからアクティブにするパッケージを選択し、[**パッケージをアクティブにする**] をクリックします。

パッケージの再ロード

開発者がサーバの実行中に Java サービスまたはフローサービスを変更した場合は、その変更を有効にするために、そのサービスを含むパッケージを再ロードする必要があります。パッケージを再ロードすると、パッケージの Java サービスを再ロードする VM クラスローダが呼び出され、フローサービスがメモリに再ロードされます。パッケージは、Designer から再ロードすることもできます。

パッケージを再ロードするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. パッケージの [**再ロード**] 列で再ロードアイコン をクリックします。
4. [**ロード済み**] 列の緑色のチェックマークアイコンは、パッケージが正常にロードされたかどうかを示します。詳細については、706 ページの「パッケージが正常にロードされたかどうかの判定」を参照してください。

パッケージの有効化

クライアントがパッケージ内のエレメントにアクセスするときは、そのパッケージが有効になっている必要があります。サーバがパッケージ内のエレメントにアクセスするには、パッケージの有効化とエレメントのロードが前提となります。デフォルトでは、パッケージは有効になっています。

無効なパッケージを有効にすると、サーバはそのパッケージ内のエレメントをメモリにロードします。

パッケージを有効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. 有効にするパッケージの [**有効**] 列の [**いいえ**] をクリックします。パッケージを有効にするかどうか、確認を促すメッセージが表示されます。[**OK**] をクリックして、パッケージを有効にします。パッケージが有効になると、[**有効**] 列に ✓ と [**はい**] が表示されます。

パッケージの無効化

パッケージ内のエレメントへのアクセスを一時的に禁止する場合は、そのパッケージを無効にします。パッケージを無効にすると、Integration Server はそのパッケージのエレメントをすべてメモリからアンロードします。パッケージ内の 1 つ以上のサービスの実行中にパッケージを無

効にすると、通常これらのサービスは失敗しますので注意が必要です。Integration Server は、処理中のサービスの終了を待機せずにパッケージを無効にします。

重要:
WmRoot パッケージは絶対に無効にしないでください。Integration Server はこのパッケージ内のサービスを使用します。

パッケージを無効にするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**パッケージ**] > [管理] に移動します。

3. 無効にするパッケージの [**有効**] 列で緑色のチェックマークをクリックします。サーバは、パッケージを無効にしてよいか確認を求めます。[**OK**] をクリックすると、パッケージが無効になります。パッケージが無効になると、[**有効**] 列に [**いいえ**] が表示されます。

   注:
   サーバは、再起動を行ってもパッケージのアクセス状態 (有効または無効) を保持します。サーバ起動時には、無効なパッケージ内のエレメントはロードされません。

パッケージの削除

パッケージ内のサービスとファイルが不要になった場合には、そのパッケージを削除できます。パッケージを削除すると、そのすべての要素 (サービス、仕様、ドキュメントタイプ) が使用できなくなります。

パッケージを削除するときは、必要に応じてそのコピーを保存します。コピーを保存すると、サーバはパッケージが *Integration Server_directory* \instances*instance_name*\replicate\salvage ディレクトリから削除される前に、そのパッケージを *Integration Server_directory* \instances*instance_name*\packages ディレクトリにコピーします。必要であれば、そのパッケージを後で回復することができます。削除したパッケージの回復については、715 ページの「パッケージの回復」を参照してください。

重要:
WmRoot パッケージは絶対に削除しないでください。Integration Server はこのパッケージ内のサービスを使用します。

コマンドプロンプトからパッケージを削除するときは、パッケージを 1 つ削除するか、または複数のパッケージのリストを削除するオプションがあります。コマンドプロンプトからパッケージを削除する方法の詳細については、73 ページの「サーバインスタンスからパッケージの削除」を参照してください。

パッケージを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**パッケージ**] > [管理] に移動します。

3. パッケージのコピーを保存しておき、後で必要になったら回復できるようにする場合は、削除するパッケージに対応する行の アイコンをオンにします。

4. コピーを保存しない場合は、削除するパッケージに対応する行の ✕ アイコンをクリックします。

5. [**削除**] をクリックします。パッケージを削除するか確認を求める画面が表示されます。[**OK**] をクリックします。

注:
削除またはバックアップ付き削除オプションを使用してパッケージを削除する場合、Integration Server は code/jars および code/jars/static ディレクトリの jar ファイルを削除しません。code/jars フォルダの jar ファイルはパッケージクラスローダによってロードされ、code/jars/static フォルダの jar ファイルは Integration Server クラスローダによってロードされます。これらの jar ファイルは Integration Server によってロックされており、Integration Server のシャットダウン時にのみ削除できます。パッケージの古いバージョン用の jar ファイルを削除せずにパッケージの新しいバージョンをインストールした場合、Integration Server は新しいパッケージバージョンで古い jar ファイルを使用します。そのため、一貫性のない動作や予期しない動作が発生する場合があります。この問題を避けるには、削除されたパッケージの新しいバージョンをインストールする前に、Integration Server をシャットダウンし、*Integration Server_directory* / instances/*instanceName*/ packages/*packageName*/code/jars および *Integration Server_directory* / instances/*instanceName*/ packages/*packageName*/code/jars/static ディレクトリに残っている jar ファイルをすべて削除します。

パッケージの回復

[**バックアップ付き削除**] オプションを使用してサーバインスタンスからパッケージを削除した後に、もう一度そのパッケージが必要になった場合は、それを回復することができます。

注:
[**バックアップ付き削除**] オプションを使用しなかった場合でも、webMethods Integration Server パッケージリポジトリから復元することはできます。パッケージがリポジトリ内に存在する場合は、コマンドプロンプトからパッケージの更新を実行できます。コマンドプロンプトからパッケージを更新する手順については、71 ページの「サーバインスタンスでのパッケージの更新」を参照してください。

[バックアップ付き削除] を使用した後でパッケージを回復するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**パッケージ**] > [管理] に移動します。

3. [**パッケージの回復**] をクリックします。

4. [**パッケージの回復**] 領域のプルダウンメニューから、回復するパッケージを選択します。

5. パッケージを回復したときに、Integration Server がパッケージを自動的にアクティブにするように設定する場合は、[**回復と同時にアクティブにする**] チェックボックスをオンにします。

6. [**回復**] をクリックします。

パッケージのアーカイブ

一般には公開されない形でパッケージをコピーする場合は、パッケージをアーカイブします。たとえば、パッケージをバックアップする場合や、パブリッシャー/サブスクライバ関係にない相手にパッケージを送る場合は、この操作を行います。

注:
バックアップを作成する別の方法には、pub.packages:backupPackage サービスを使用する方法があります。ただし、backupPackage サービスを使用する場合、バックアップされたパッケージのパッケージメタデータは元のパッケージと同じです。たとえば、作成タイムスタンプは、元のパッケージの作成タイムスタンプを反映します。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

パッケージをアーカイブするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**パッケージ**] > [管理] に移動します。

3. [**パッケージの一覧**] からアーカイブするパッケージを検索して、 アイコンをクリックします。

   サーバ上に表示されたページで、アーカイブするファイル、アーカイブのタイプ (フルまたはパッチ) およびバージョン情報を指定します。情報の指定については、729 ページの「リリースまたはアーカイブに関するファイルおよびバージョン情報の指定」を参照してください。

   注:
   パッケージのインストール後に、パッケージを自動的にアーカイブすることもできます。詳細については、739 ページの「別のサーバからパブリッシュされたパッケージのインストールについて」を参照してください。

サーバ間でのパッケージのコピー

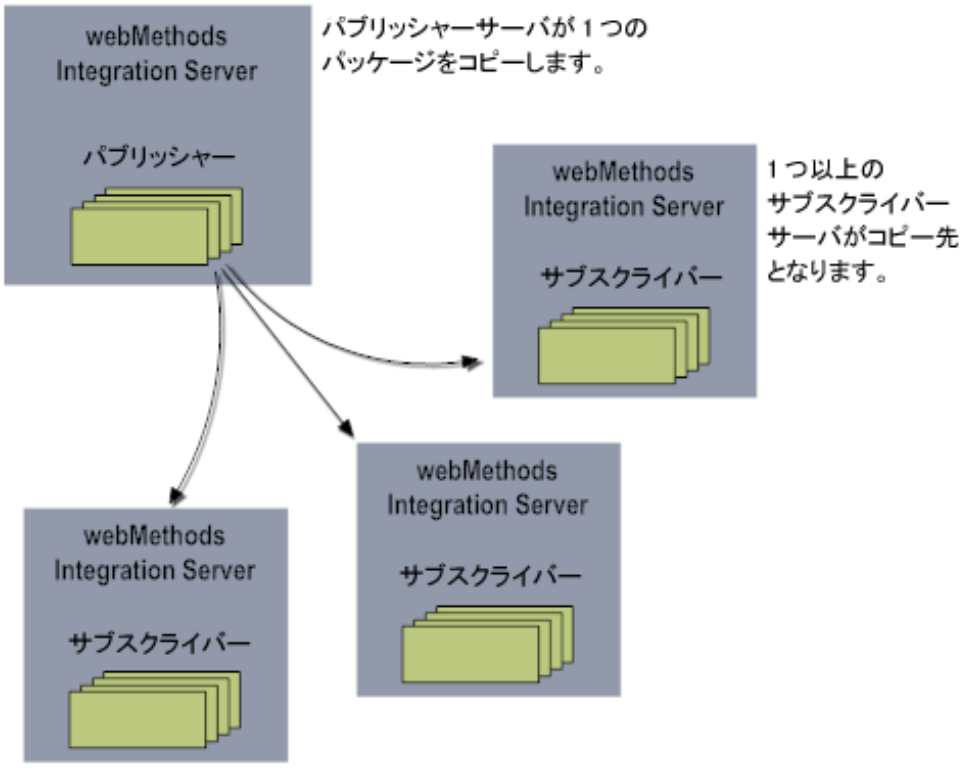
ある Integration Server から別のサーバにパッケージをコピー (パブリッシュ) するには、パッケージの複製を使用します。クラスタ環境ではこの機能を利用して、クラスタ内のすべてのサーバに対する新規パッケージや更新されたパッケージの複製を迅速に行えます。また、この機能は、1 つのサーバから Web 上の別のサーバにパッケージを配信する場合にも役立ちます。

注:
Software AG Designer を使用すると、コピー操作またはドラッグアンドドロップ操作を行って、パッケージとその内容を別の Integration Server にコピーすることができます。Designer を使用してパッケージをコピーすると、リモート環境などで一連のサービスとそれらのサービスがサポートするファイルをすばやくテストすることが可能です。この方法は、変更管理が重要ではない単一の開発環境の場合に便利です。ただし、実稼動環境では、パッケージの複製機能を使用することをお勧めします。

注:
パッケージのコピーを作成するだけで、それを別のサーバに送信しない場合、たとえばそのバックアップをとりたい場合は、716 ページの「パッケージのアーカイブ」を参照してください。

パッケージの複製の概要

複製処理中には、単一の Integration Server が 1 つ以上の受信側サーバに指定のパッケージを送信 (パブリッシュ) します。パッケージの送信元のサーバはパブリッシャーと呼ばれ、送信先のサーバはサブスクライバと呼ばれています。



サブスクライバサーバは、受信ディレクトリ (*Integration Server_directory* \instances*instance_name*\replicate\inbound) でパッケージを受信します。新しいパッケージをアクティブ化するために、サブスクライバサーバの管理者はパッケージ到着後にそれをインストールする必要があります。(この手順は、739 ページの「別のサーバからパブリッシュされたパッケージのインストールについて」に記載されています)。

サブスクリプションは、パブリッシャーまたはサブスクライバが要求します。パブリッシャーはパッケージを送信 (プッシュ) でき、サブスクライバはパッケージを要求 (プル) することができます。

パッケージを別のサーバに送信するときは、前もって**リリース**を作成する必要があります。リリースが作成されると、サーバはパッケージおよびパッケージに関する情報を含む配布ファイルを作成し、サブスクライバ側でそのパッケージを使用できるようにします。

リリースは、所定パッケージに対して複数作成できます。たとえば、あるパッケージのバージョン 1.0、1.1 および 1.2 に対して個別のリリースを作成することができます。あるいは、各対象ユーザの個々のパッケージに対して異なるリリースを使用できます。各リリースには固有の名前を指定する必要があります。

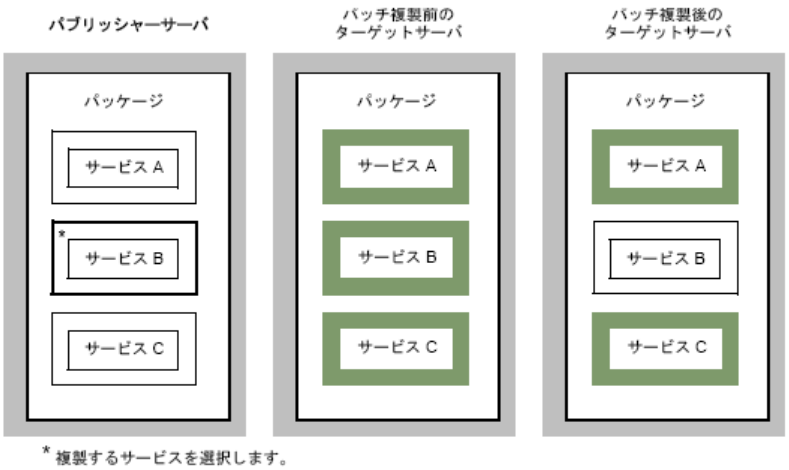
重要:
ある特定のパッケージに複数のリリースが使用されている場合に、1 つ以上のサブスクライバが自動プル機能を指定していると、その新規リリースが使用可能になった時点で、サブスクライバはそのパッケージのすべてのリリースを受信します。自動プル機能の詳細については、732 ページの「サブスクライバサーバ」を参照してください。

リリースには完全なパッケージを含めるか (フルリリース)、またはパッケージのパッチのみを含むことができます (パッチリリース)。通常、パッケージに大きな変更を加えたときはフルリリースをパブリッシュし、パッケージの問題を修正するときだけパッチを使用します。

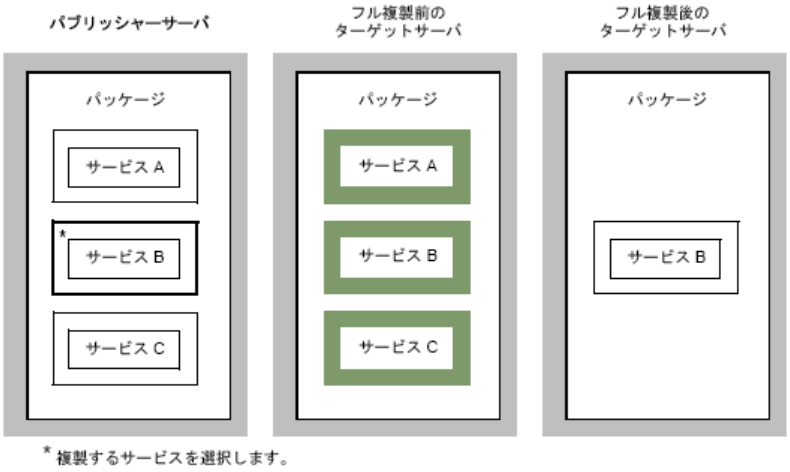
フルリリースでは、サブスクライバのサーバ上の旧パッケージが新しいパッケージに完全に置き換えられます。パッチリリースでは、パッチリリース内のファイルがターゲットパッケージ内の同バージョンのファイルに取って代わりますが、パッケージ内の他のファイルはすべてそのまま残ります。

フルリリースまたはパッチリリースの指定以外にも、リリースに入るすべてのファイルまたは一部のファイルを選択することができます。

次の図は、パッチリリースによるファイルの置き換えを示したものです。

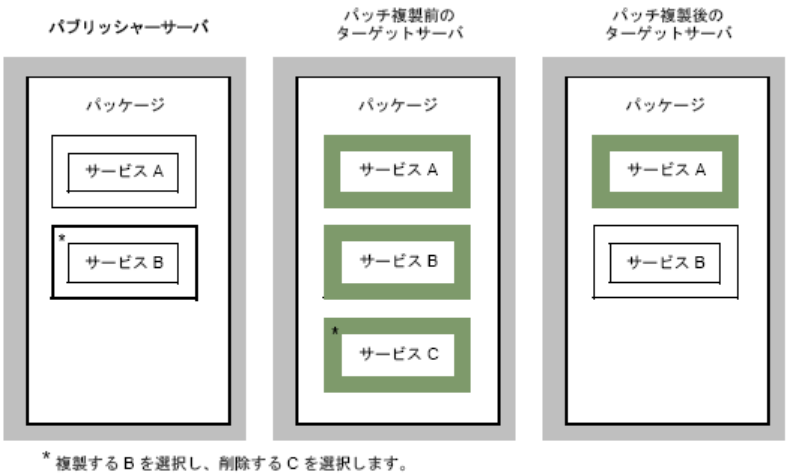


次の図では、複製のために単一のサービスが選択され、前の場合とは違ってフルリリースが指定されています。

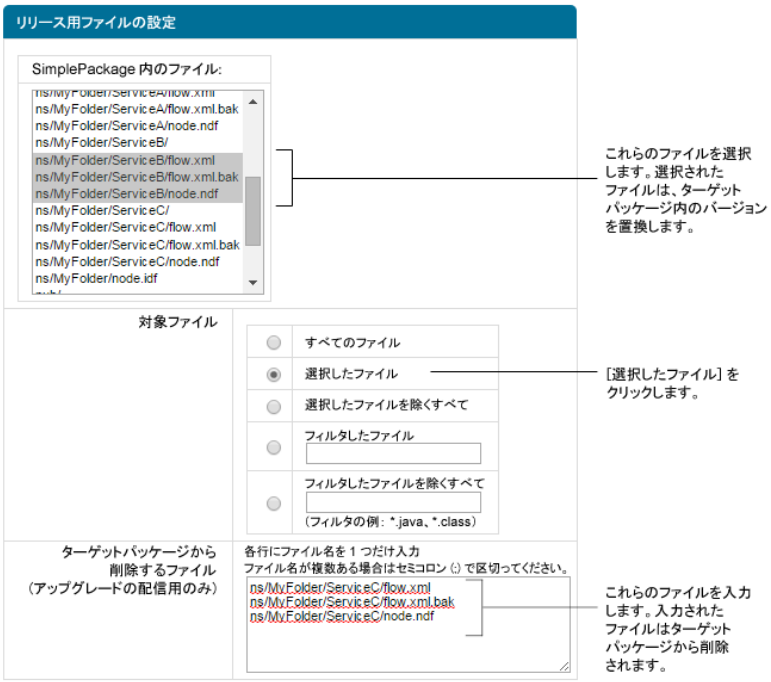


たいていの場合は、すべてのファイルを選択してフルリリースを指定するか、一部のファイルを選択してパッチリリースを指定することになります。しかし、一部のファイルだけを選択してフルリリースを選択する場合もあり得ます。たとえば、他の人には公表したくないファイル (内部文書のファイルなど) がパッケージに入っている場合も考えられます。余分なファイルを除くすべてのファイルを選択してフルリリースを指定すれば、必要なファイルだけを含むパッケージが出来上がります。

また、一部のファイルは置き換え、別の一部のファイルはそのまま残し、それ以外のファイルは削除する場合もあるでしょう。こういった細かい処理を行う場合は、パッチリリースを実行し、コピーするファイルと削除するファイルを指定します。コピーや削除を指定しないファイルはそのまま残ります。次の例では、サービス A をそのまま残し、サービス B を置き換え、サービス C をターゲットパッケージから削除します。



このタスクを行うには、次の図のように [**リリース用のファイルの設定**] ページでの指定が必要になります。



パッケージ、Integration Server、JVM のそれぞれのバージョンは Integration Server によって追跡管理されます。これにより、インストール中のパッケージがサブスクライバサーバの環境と適合していることが確認されます。バージョンのチェック方法は、リリースがフルリリースかパッチリリースかで異なります。

注:
パッチリリースがパッケージに適用されている場合、開発者は Designer でパッケージを表示するときにパッチ履歴を確認することができます。しかし、パブリッシャーがパッケージのフルリリースをパブリッシュすると、パッチ履歴は削除されます。

バージョンのチェック

サブスクライブするサーバの管理者がパッケージをインストールすると、サブスクライブするサーバはバージョンを確認します。

ターゲットサーバによる確認

ターゲット JVM のバージョン

ターゲットサーバは、リリースの作成時に指定された JVM バージョンと同じかまたはそれ以降のバージョンを実行しています。この条件が満たされていないと、サブスクライバサーバは警告を発してパッケージをインストールしますが、そのパッケージを非アクティブのままにします。

ターゲット Integration Server は、パッケージで必要な JVM バージョンと同じかそれ以降のメジャーバージョンの JVM バージョンで実行さ

ターゲットサーバによる確認

	れている必要があります。たとえば、パッケージリリースの指定 JVM バージョンが 1.8.0_32 である場合、インストール先 Integration Server は 1.8 以降の JVM で実行されている必要があります。Integration Server JVM のメジャーバージョンがパッケージで必要な JVM より以前のバージョンである場合、Integration Server はパッケージをインストールしますが、アクティブにしません。 パッケージのアクティブ化については、712 ページの「パッケージのアクティブ化」を参照してください。	
パッケージのバージョン	**フルリリースの場合**	**パッチリリースの場合**
	ターゲットサーバ上のパッケージのバージョンは、インストールされるパッケージと同じかそれ以前のバージョンです。この条件が満たされていないと、パッケージのインストールは失敗します。	ターゲットサーバ上のパッケージのバージョンは、(リリース作成中に指定された) リリースに必要なバージョンとまったく同じです。この条件が満たされていないと、パッケージのインストールは失敗します。
	たとえば、新しいリリースを作成して、そのリリースにバージョン 2.0 の wmExample パッケージが含まれるように指定する場合には、ターゲットシステム上の wmExample はリリース 2.0 かそれ以前のものである必要があります。	たとえば、wmExample パッケージバージョン 2.0 のパッチを含む新規リリースを作成し、ターゲットパッケージがバージョン 2.0 になるよう指定すると、ターゲットバージョンが 2.0 でないとパッケージのインストールは失敗します。
	この制限によって、うっかり旧バージョンのパッケージを新しいバージョンにインストールするような操作ミスを防ぐことができます。	この制限により、パッチの当て方や当て先が制御しやすくなります。パッチはリリースに依存しているのが普通なので、この制限はかなり有効です。

サブスクライブできるサーバ

送信側と受信側のサーバにセキュリティ上の問題がない限り、任意の Integration Server が別のサーバ上のパッケージをサブスクライブできます。パッケージ複製のためのセキュリティは、以下のようなさまざまな方法で実現することができます。

- **ユーザ ID およびパスワード**パッケージをサブスクライバに送信する場合、パブリッシャーはサブスクライバ上にあるユーザ ID とパスワードを指定してサブスクライバにログインしなければなりません。
- **ACL**サブスクライバにログオンするためにパブリッシャーが使用するユーザ ID は、サブスクライバ上の Replicators ACL かそれより上位の ACL に割り当てられるグループのメンバーでなければなりません。

- **SSL** SSL を使用してパッケージの複製に関わるサーバが互いに接続されるように指定できます。

パブリッシャーは、パッケージごとにサブスクライバサーバのリストを保持しています。

サブスクリプションは、以下のようにパブリッシャーまたはサブスクライバによって追加できます。

- **パブリッシャー**パブリッシャーサーバの管理者は、パブリッシャーの機能を使用して、パブリッシャーサーバを送信元とする任意のパッケージ (サブスクライブしないパッケージ) にサブスクライバを追加 (または削除) することができます。
- **サブスクライバ**リモート Integration Server (サブスクライバ) の管理者は、パブリッシャーにサブスクリプション要求を送信できます。パブリッシャーはこの要求を受信すると、認証が成功している場合に限り、要求されたパッケージのサブスクリプションリストにそのサーバを自動的に追加します。サブスクライバは、サブスクライブ対象のパッケージについて取り消し要求を発行する (サブスクリプションを取り消す) こともできます。

パッケージ複製機能の使用に関するガイドライン

パッケージ複製機能を使用する場合は、次のガイドラインに留意してください。

- パブリッシャーと参加サブスクライバは、Integration Server バージョン 2.0 以上を使用する必要があります。自動プル機能を使用する場合は、バージョン 4.0 以上が必要となります。バージョン 4.0 以上の Integration Server を実行してそれより前のリリースの Integration Server にパブリッシュしても、サブスクライバはパッケージの手動プルも自動プルも実行できません。サブスクライバは、パブリッシャーがパッケージを送信するのを待たねばなりません。
- 任意の Integration Server はパッケージをパブリッシュできます。
- Integration Server はすべて、別の Integration Server 上のパッケージにサブスクライブできます。
- Integration Server はパッケージのパブリッシャーと他のパッケージのサブスクライバであることはできますが、同じパッケージのパブリッシャーとサブスクライバを兼ねることはできません。
- サブスクリプションの設定後に、その設定を行ったユーザアカウント (パブリッシャーサーバがログオンのために使用するサブスクライバサーバ上のアカウント) を削除すると、パブリッシャーはサブスクライバサーバにログインしてこのパッケージを送信することができなくなります。
- ポートに関連付けられているパッケージを複製すると、複製先のシステムではその新しいポートによってセキュリティが低下する恐れがあるので注意してください。たとえば、HTTP ポート 5556 に関連付けられているパッケージを複製する場合は、複製プロセスによって複製先のサーバに HTTP ポート 5556 が作成されます。セキュリティの高さを理由に複製先のサーバで通常 HTTPS ポートだけを使用している場合は、この新規ポートがサーバ上のセキュリティホールになる可能性があります。

パブリッシャーサーバ

ここでは、使用中のサーバがパブリッシャーサーバとしてパッケージの複製に参加するときに実行するタスクについて説明します。

タスク	参照先
パッケージのサブスクライバのリストの表示	723ページの「特定のパッケージのサブスクライバの表示」
パッケージのサブスクライバの指定	724ページの「パブリッシャーサーバのサブスクライバの追加」
サブスクライバ情報の更新	725 ページの「サブスクライバ情報の更新」
パッケージのサブスクライバの削除	726ページの「パッケージのサブスクライバの削除」
サブスクライバサーバへのパッケージのパブリッシュ	727 ページの「パッケージのパブリッシュ」
リリースまたはアーカイブに関するファイルおよびバージョン情報の指定	729 ページの「リリースまたはアーカイブに関するファイルおよびバージョン情報の指定」

特定のパッケージのサブスクライバの表示

サーバ上の特定のパッケージのサブスクライバのリストを表示するには、以下の手順に従います。

単一パッケージのサブスクライバを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. サブスクライバを表示するパッケージの名前をクリックします。

   サーバは、[**サブスクライバ**] フィールドにパッケージのサブスクライバを一覧します。

すべてのパッケージのサブスクライバの表示

サーバ上のすべてのパッケージのサブスクライバのリストを表示するには、以下の手順に従います。

すべてのパッケージのサブスクライバを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**パッケージ**] > [パブリッシュ] に移動します。

   すべてのパッケージ、そのサブスクライバおよびリリースのリストが表示されます。

パブリッシャーサーバのサブスクライバの追加

サブスクライバを追加するときは、パッケージを受信する Integration Server を指定します。サーバ上では、パッケージごとに異なるサブスクライバリストを保持できます。

パッケージのサブスクライバ (受信側) を指定します(この作業が必要となるのは、パッケージを最初にパブリッシュするときだけです。それ以降は、最初のリストを変更または再利用するだけで済みます)。

パブリッシャーサーバのサブスクライバを追加するときは、次の点に留意します。

- 自動プル機能を指定する場合は、サブスクライバからサブスクリプションを作成する必要があります。
- サブスクライバを追加するときは、サブスクライバサーバが稼動している必要があります。

> **注:**
> サブスクライバサーバからサブスクリプションを要求する場合は、734 ページの「サブスクライバサーバのパッケージのサブスクライブ」を参照してください。

サブスクライバを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [パブリッシュ] に移動します。
3. [**サブスクライバの追加**] をクリックします。
4. [**パッケージ**] フィールドのドロップダウンリストからサブスクライバを識別するためのパッケージを選択します。
5. サブスクライバサーバを指定するには、次のフィールドに情報を入力します。

フィールド	説明
[ホスト名]	サブスクライバサーバが稼動しているマシンの名前。
[ホストポート]	サブスクライバサーバが、パブリッシュされるこのパッケージを受信待機するポート番号。
[転送手段]	パブリッシャーサーバがパッケージをサブスクライバサーバに送信するための手段。HTTP または HTTPS を選択します。デフォルトは HTTP です。 **注:**

フィールド	説明
	パッケージをサブスクライバに送信するときにパブリッシャーに SSL を使用させたい場合は、ここで HTTPS を指定します。 パブリッシャーは、サブスクライバに接続する場合、パブリッシャーの [**セキュリティ**] > [認証] ページで指定されているサーバのデフォルトの [**送信 SSL 認証**] を使用します。
[リモートユーザ名]	パブリッシャーサーバがサブスクライバサーバにログインするときに使用するユーザ。このユーザは、サブスクライバサーバ上の Replicators ACL に割り当てられるグループのメンバーである必要があります。
[リモートパスワード]	パブリッシャーサーバがサブスクライバサーバにログインするときに使用するユーザのパスワード。
[通知先電子メールアドレス]	パブリッシャーサーバがパッケージをいつリリースするか通知するための管理者の電子メールアドレス。

6. [**サブスクライバの追加**] をクリックします。サーバは、[**サブスクライバ**] フィールドのリストにサブスクライバを追加します。

   パッケージのサブスクライバとして指定するサーバごとにこの手順を繰り返します。

サブスクライバ情報の更新

パッケージ名などのサブスクライバに関する情報を更新するには、以下の手順に従います。

サブスクライバ情報の更新

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [パブリッシュ] に移動します。
3. [**サブスクライバの更新と削除**] をクリックします。
4. サブスクライバ情報リストでこのサブスクライバを検索して、[**更新**] 列の [**編集**] をクリックします。
5. サブスクライバ情報を変更するために、下記のフィールドに必要な情報を入力します。

フィールド	説明
[パッケージ]	サブスクライバがサブスクライブするパッケージ。サブスクリプションは、別のパッケージ用に変更できます。同じパッケージのパブリッシュとサブスクライブを同時に行うことはできないので、パッケー

フィールド	説明
	ジについては、サーバがまだサブスクライブしていないものしか選択できません。
[ホスト名]	サブスクライバサーバが稼動しているマシンの名前。
[ホストポート]	サブスクライバサーバが、パブリッシュされるこのパッケージを受信待機するポート番号。指定する番号は、サブスクライバサーバ上で有効になっている既存のポートに対応している必要があります。また、パブリッシャーサーバは、Replicator かそれより上位のものにアクセスできる必要があります。
[転送手段]	パブリッシャーサーバがパッケージをサブスクライバサーバに送信するための手段。HTTP または HTTPS を選択します。デフォルトは HTTP です。転送手段のタイプは、サブスクライバサーバ上のホストポートについて定義されたタイプと同じである必要があります。 **注:** パッケージをサブスクライバに送信するときにパブリッシャーに SSL を使用させたい場合は、ここで HTTPS を指定します。 パブリッシャーは、サブスクライバに接続する場合、パブリッシャーの [**セキュリティ**] > [認証] ページで指定されているサーバのデフォルトの [**送信 SSL 認証**] を使用します。
[リモートユーザ名]	パブリッシャーサーバがサブスクライバサーバにログインするときに使用するユーザ。このユーザは、サブスクライバサーバ上の Replicators ACL に割り当てられるグループのメンバーである必要があります。
[リモートパスワード]	パブリッシャーサーバがサブスクライバサーバにログインするときに使用するユーザのパスワード。
[通知先電子メールアドレス]	パブリッシャーサーバがパッケージをいつリリースするか通知するための管理者の電子メールアドレス。

6. [**変更内容の保存**] をクリックします。サーバは、[**サブスクライバ**] フィールドのリストにサブスクライバを追加します。サーバは、サブスクライバサーバとパブリッシャーサーバの両方の情報を更新します。

パッケージのサブスクライバの削除

パブリッシュするパッケージからサブスクライバを削除するには、以下の手順に従います。

注:
サブスクライバがパブリッシャーによって開始されたサブスクリプションを削除すると、サブスクライバサーバはサブスクリプションリストからそのサブスクリプションを削除しますが、そのサブスクリプションがパブリッシャーのリストからすぐに削除されるわけではありません。次回、パブリッシャーサーバがサブスクライバにパッケージを送信しようとすると、

パブリッシャーにその削除が通知され、パブリッシャーのリストからサブスクリプションが削除されます。

サブスクライバの削除

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [パブリッシュ] に移動します。
3. [**サブスクライバの削除**] をクリックします。
4. サブスクライバを削除するパッケージを検索して、[**削除**] フィールドのボックスをオンにします。

注:
サブスクライバリストから実行中のサブスクライバを削除すると、パブリッシャーはサブスクライバにその削除を知らせます。しかし、それが実行中のサブスクライバでない場合は、サブスクリプションの取り消しがサブスクライバに通知されません。この場合は、サブスクライバサーバが使用可能になった時点でそのサーバからサブスクリプションを手動で削除します。

パッケージのパブリッシュ

パッケージをほかのIntegration Server にパブリッシュする場合は、次の 2 つのタスクを実行します。

- **リリースの作成**パッケージをパブリッシュするために、サーバはパッケージの情報を含む配布ファイルを作成します。

  配布ファイルを作成するときは、ファイルに入れる情報を選択します。

  送信するファイルは、全部または一部だけを選択できます。また、フルリリースかパッチリリースを要求することもできます。 フルリリースでは、サブスクライバのサーバ上の旧パッケージが新しいパッケージに完全に置き換えられます。パッチリリースでは、パッチリリース内のファイルがターゲットパッケージ内の同バージョンのファイルに取って代わりますが、パッケージ内の他のファイルはすべてそのまま残ります。フルリリースとパッチリリースの違いの詳細については、717 ページの「パッケージの複製の概要」を参照してください。

  リリースに入るファイルが指定されると、サーバはその選択されたファイルをすべて単一の圧縮ファイル (圧縮ファイル) に収容します。その圧縮ファイルは、*Integration Server_directory* \instances*instance_name*\replicate\outbound ディレクトリ内に置かれます。このパッケージの圧縮ファイルが outbound ディレクトリ内に既に存在する場合は、その圧縮ファイルに上書きされます。

- **リリースの送信**リリースを作成後、それをサブスクライブするサーバに送信できます。

  サブスクライバサーバは、リリースを含む圧縮ファイルを受信して inbound ディレクトリ (*Integration Server_directory* \instances*instance_name*\replicate\inbound) に収めます。パッケージの圧縮ファイルがサブスクライバサーバの inbound ディレクトリ内に既に存在する場

合は、その圧縮ファイルに上書きされます。圧縮ファイルは、そのサーバの管理者がパッケージをインストールするまで inbound ディレクトリ内に保持されます。

開発者は、リリース作成時にサービスを実行するパッケージを設定できます。リリース作成を始めると、このサービスが実行され、圧縮されるファイルのリストが表示されます。このサービスを使用して、パッケージの状態情報や設定情報をファイルに書き込むことができます。このファイルは、リリース内の他の圧縮ファイルに含まれます。複製サービスを設定する手順については、『*webMethods Service Development Help*』を参照してください。

重要:
パッケージをパブリッシュする場合は、前もってサブスクリプションを指定する必要があります。詳細については、724 ページの「パブリッシャーサーバのサブスクライバの追加」を参照してください。

リリースの作成

パッケージのリリースを作成するには、以下の手順に従います。

リリースを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [パブリッシュ] に移動します。
3. [**リリースの作成と削除**] をクリックします。
4. リリースを作成するパッケージを検索して、[**リリースの作成**] をクリックします。
5. サーバ上に表示されたページで、リリースに入れるファイル、リリースのタイプ (フルまたはパッチ) およびバージョン情報を指定します。情報の指定については、729 ページの「リリースまたはアーカイブに関するファイルおよびバージョン情報の指定」を参照してください。

リリースの送信

パッケージリリースを送信するには、以下の手順に従います。

リリースを送信するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [パブリッシュ] に移動します。
3. [**使用可能なリリース**] セクションで、送信するパッケージのリリースを検索して、[**リリースの送信**] をクリックします。

リリースまたはアーカイブに関するファイルおよびバージョン情報の指定

パッケージをアーカイブしたりリリースを作成したりするときは、アーカイブまたはリリースするファイル、アーカイブまたはリリースのタイプ (フルまたはパッチ) およびバージョン情報をサーバ上のページから指定できます。

注:
また、watt.server.createPackage.ignorePattern 設定パラメータを使用して、パッケージをパブリッシュするときに除外するファイルのファイル名のパターンを指定することもできます。このパラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

パッケージをアーカイブする方法の詳細については、716 ページの「パッケージのアーカイブ」を参照してください。さらに、リリースを作成および送信する方法の詳細については、727 ページの「パッケージのパブリッシュ」を参照してください。

リリースまたはアーカイブされたパッケージのファイルおよびバージョン情報を指定するには

1. リリースまたはアーカイブに入れるファイルを指定します。

リリースまたはアーカイブに入れるファイル	手順
すべてのファイル	[**対象ファイル**] セクションの [**すべてのファイル**] を選択します。
ほとんどのファイル	[*package* **内のファイル**] セクションで、アーカイブまたはリリースに入れないファイルを選択します。 [**対象ファイル**] セクションで [**選択したファイルを除くすべて**] を選択します。 最終アーカイブまたは最終リリースが作成された後に、開発者がパッケージの依存、開始サービス、シャットダウンサービスまたは複製サービスをパッケージに追加している場合は、必ず manifest.v3 ファイルを追加してください。そうしないと、これらのサービスは生成されたパッケージ内で使用できなくなります。開始サービス、シャットダウンサービスおよび複製サービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
ごく少数のファイルのみ	[*package* **内のファイル**] セクションで、アーカイブまたはリリースに入れるファイルを選択します。 [**対象ファイル**] セクションの [**選択したファイル**] を選択します。 最終アーカイブまたは最終リリースが作成された後に、開発者がパッケージの依存、開始サービス、シャットダウンサービスまたは

リリースまたはアーカイブに入れるファイル	手順
	複製サービスをパッケージに追加している場合は、必ず manifest.v3 ファイルを追加してください。そうしないと、これらのサービスは生成されたパッケージ内で使用できなくなります。開始サービス、シャットダウンサービスおよび複製サービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
類似したパス名を持つファイル	[**対象ファイル**] セクションの [**フィルタしたファイル**] を選択し、有効な値 (たとえば *.java または *.class) を入力します。 または 類似名を持つファイル以外のすべてのファイルを取り込むために、[**対象ファイル**] セクションで [**フィルタしたファイルを除くすべて**] をクリックし、有効な値 (たとえば *.bak) を入力します。 最終アーカイブまたは最終リリースが作成された後に、開発者がパッケージの依存、開始サービス、シャットダウンサービスまたは複製サービスをパッケージに追加している場合は、必ず manifest.v3 ファイルを追加してください。そうしないと、これらのサービスは生成されたパッケージ内で使用できなくなります。開始サービス、シャットダウンサービスおよび複製サービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。 次の特殊文字を指定してパターン照合を行うことができます。 **文字** / **説明** / **例** * / 任意の数の文字に対応 / *.java

2. [**ターゲットパッケージから削除するファイル**] フィールドで、ターゲットパッケージから削除するファイルを指定します。各エントリはセミコロン (;) で区切ります。サブスクライバサーバがパッケージをインストールすると、これらのファイルがターゲットパッケージから削除されます。

3. パッケージのバージョン情報と説明を指定します。

フィールド	説明
[**アーカイブタイプ**] または [**リリースタイプ**]	[**フル**]: アーカイブまたはリリースに書き込まれるパッケージ内のすべてのファイル。 [**パッチ**]: アーカイブまたはリリースに書き込まれるパッケージ内の選択されたファイル。ターゲットサーバ上の管理者がパッチアーカイブまたはパッチリリースをインストールすると、そのパッチアーカイブまたはパッチリリースに含まれるファイルがターゲッ

フィールド	説明
	トパッケージ内のそのバージョンのファイルに取って代わります。ターゲットパッケージ内の他のファイルは、すべて元のままの状態で残ります。 最終アーカイブまたは最終リリースが作成された後に、開発者がパッケージの依存、開始サービス、シャットダウンサービスまたは複製サービスをパッケージに追加している場合は、必ず manifest.v3 ファイルを追加してください。そうしないと、これらのサービスは生成されたパッケージ内で使用できなくなります。開始サービス、シャットダウンサービスおよび複製サービスの詳細については、757 ページの「パッケージのロード、アンロードまたは複製時に実行するサービス」を参照してください。
[アーカイブ名] または [リリース名]	アーカイブまたはリリースに割り当てる名前 (WmExample パッケージのベータリリースなど)。
[簡単な説明]	アーカイブまたはリリースに割り当てる説明。たとえば、「OrderProcess の問題を修正するためのパッチを含む Dec リリース」。
[バージョン]	アーカイブまたはリリースするパッケージに割り当てるバージョン番号。このバージョンは、パッケージ自身のバージョンとは同じでない場合があります。開発者が初めてパッケージを作成すると、バージョン番号は 1.0 に設定されます。 Integration Server の動作チェックの詳細については、720 ページの「バージョンのチェック」を参照してください。
[ビルド番号]	パッケージを生成するたびに開発者がそのパッケージに割り当てる番号。たとえば、バージョン 1.0 の WmExample パッケージを 10 回生成する場合に、開発者はその都度ビルド番号を 1、2、3...10 と割り当てます。
[含まれているパッチ]	このリリースのパッケージに適用されたパッチのリスト。これらはインストールにとって重要な番号であり、障害追跡システムから取得できます。

4. サブスクライバの設定を指定します。

フィールド	意味
webMethods Integration Server	ターゲットサーバ上で実行する必要がある webMethods サーバのバージョン。 サブスクライバサーバで実行されるバージョンチェックの詳細については、720 ページの「バージョンのチェック」を参照してください。

フィールド	意味
[JVM バージョンの最小値]	ターゲットの Integration Server がこのパッケージを使用するときに実行する必要がある JVM の最小バージョン。管理者がパッケージをインストールすると、サーバは実行中の JVM のバージョンをチェックします。JVM の以前のメジャーバージョンを実行していれば、サーバはパッケージをインストールしますが、それをアクティブにはしません。 サブスクライバサーバで実行されるバージョンチェックの詳細については、720 ページの「バージョンのチェック」を参照してください。

5. ターゲットパッケージのバージョンを指定します (パッチリリースの場合のみ)。

6. これは、ターゲットサーバが実行する必要があるパッケージのバージョンです。管理者がパッチをターゲットサーバにインストールすると、サーバはターゲットパッケージのバージョンがここで指定されたものと同じかどうかを確認します。ターゲットパッケージのバージョンが違っていれば、そのパッケージはインストールされません。この制限により、パッチの当て方や当て先が制御しやすくなります。パッチはリリースに依存しているのが普通なので、この制限はかなり有効です。

サブスクライバサーバ

ここでは、サブスクライバサーバがパッケージの複製に参加するときに実行するタスクについて説明します。

サブスクライバは、パッケージを手動で、または自動的に抽出できます。パッケージを手動で抽出する場合、サブスクライバサーバ上の管理者は使用可能なサブスクリプションのリストを表示して、必要なパッケージを抽出します。自動プルが有効な場合、サブスクライバサーバは、新規リリースが使用可能になった時点でパッケージをパブリッシャーから自動的にプルします。

パッケージを自動的に抽出する場合、サブスクライバはサブスクリプションを設定するときに自動プル機能を指定する必要があります。新規リリースが使用可能になると、パブリッシャーサーバはサービス呼び出しの電子メールを指定の電子メールサーバに送信します。サービス呼び出し電子メールには、パッケージの抽出のためにサブスクライバサーバ上で実行されるサービスの呼び出しが含まれています。サブスクライバサーバは、内部の電子メールポートを介して電子メールサーバを定期的にチェックします。サブスクライバサーバは、サービス呼び出し電子メールを受信して処理すると、パブリッシャーからパッケージを自動的にプルして、それを Inbound ディレクトリに置きます。サブスクライバサーバ上の管理者は、そのパッケージをインストールすることができます。

タスク	参照先
使用サーバがサブスクライブするパッケージの表示	733 ページの「使用サーバがサブスクライブするパッケージの表示」
パッケージの手動プル	733 ページの「パッケージの手動プル」

タスク	参照先
別のサーバからのパッケージのサブスクライブ	734 ページの「サブスクライバサーバのパッケージのサブスクライブ」
サブスクリプション情報の更新	725 ページの「サブスクライバ情報の更新」
別のサーバ上のパッケージに対するサブスクリプションのキャンセル	739ページの「サブスクライブのキャンセル」
別のサーバからパブリッシュされたパッケージのインストール	739ページの「別のサーバからパブリッシュされたパッケージのインストールについて」

使用サーバがサブスクライブするパッケージの表示

使用サーバがほかのサーバ上のパッケージに対して持つサブスクリプションを表示することができます。

使用サーバがサブスクライブするパッケージを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [サブスクライブ] に移動します。
3. 使用可能なサブスクリプションが、パブリッシャー、パッケージ、リリースごとに整理されて一覧表示されます。
4. サーバは、ユーザ (サブスクライバ) がサブスクリプションを追加、更新または削除すると、この情報を自動的に更新します。ただし、パブリッシャーによる変更を確認するには、[**すべてのサブスクリプションの詳細を更新**] をクリックする必要があります。

パッケージの手動プル

使用中のサーバの inbound ディレクトリにサブスクリプションを手動でプルすることができます。

既にサブスクライブしているパッケージをプルするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [サブスクライブ] に移動します。
3. 使用可能なサブスクリプションが、パブリッシャー、パッケージ、リリースごとに整理されて一覧表示されます。
4. プルするパッケージのリリースを検索して、[**抽出**] フィールドで特定の抽出方法をクリックします。

フィールド	意味
[サービス呼び出し経由]	パブリッシャーサーバは、HTTP または HTTPS を使用してリリースを送信します。
[FTP 経由でダウンロード]	パブリッシャーサーバは、FTP を使用してリリースを送信します。 FTP を選択すると、FTP を使用するための以下の情報を入力するよう要求されます。 [**リリース名**]: リリースに割り当てられる名前。たとえば、WmExample パッケージのベータリリース。 [**リモートサーバエイリアス**]: パブリッシャーサーバが常駐するマシンの名前。 [**リモートサーバ FTP ポート**]: パブリッシャーがパッケージの送信に使用するパブリッシャーサーバ上の FTP ポート。 [**リモートユーザ名**]: サブスクライバがパブリッシャーサーバにログインするときに使用するユーザ。 [**リモートパスワード**]:サブスクライバサーバがサブスクライバサーバにログインするときに使用するユーザのパスワード。

5. パッケージをインストールします。手順については、739 ページの「別のサーバからパブリッシュされたパッケージのインストールについて」を参照してください。

サブスクライバサーバのパッケージのサブスクライブ

サブスクライバサーバからパッケージをサブスクライブすると、使用中のサーバはサブスクリプション要求をパブリッシャーサーバに送ります。パブリッシャーサーバは、パッケージのサブスクリプションリストにその要求元のサーバを追加します。

リモートサーバは、ローカルサーバ上で定義されたエイリアスを持っている必要があります。リモートサーバのエイリアスが定義されていない場合は、ナビゲーションパネルの [**外部サーバ**] > [リモートサーバ] に移動してエイリアスを事前に定義するか、サブスクリプションを作成するときに定義します。

サブスクライバは、サブスクリプションを要求するときに、次の両方向の接続情報をパブリッシャーに提供する必要があります。

- **サブスクライバがパブリッシャーに接続してサブスクリプション要求を行うための方法**サブスクライバは、パブリッシャーサーバにログオンするための有効なユーザ ID とパスワードを指定しなければなりません。接続情報を設定するときは、パブリッシャーのリモートサーバエイリアスを使用します。
- **パブリッシャーがサブスクライバに接続してパッケージを送信するための方法**サブスクライバは、パブリッシャーがサブスクライバサーバにログオンするための有効なユーザ ID とパスワードを与えなければなりません。ユーザ ID は、Replicators ACL に割り当てられるグ

ループのメンバーである必要があります。サブスクライバは、受信待機ポートなど、他の接続情報も提供する必要があります。

別のサーバからのパッケージのサブスクリプション要求

次に示す手順は、サブスクライブの要求方法に関するものです。

注:
サブスクリプションを追加するときは、パブリッシャーサーバが稼動している必要があります。

別のサーバからのパッケージのサブスクリプションを要求するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [サブスクライブ] に移動します。
3. [**リモートパッケージのサブスクライブ**] をクリックします。
4. [**パッケージ**] フィールドにパッケージ名を入力します。パッケージ名は、パブリッシャーサーバで指定したように大文字と小文字の組み合わせもそのまま使い分けて入力してください。
5. 次のフィールドに情報を入力して要求を設定します。

フィールド	説明
[パブリッシャーのエイリアス]	パブリッシャーに割り当てるエイリアス。サブスクライバは、サブスクリプション登録のために必要なパブリッシャーサーバとの接続方法をエイリアスの定義から知ることができます。エイリアスには、ホスト名または IP アドレスなどの接続情報が含まれています。このパブリッシャーのエイリアスを定義していない場合は、[リモートサーバ] ページへのリンクをクリックします。このページで、パブリッシャーのエイリアスを設定できます。詳細については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。
[ローカルポート]	サブスクライバがパブリッシャーから送信されるパッケージを受け取るための、受信待機ポートの番号。この設定によって、パブリッシャーの送信手段が HTTP になるか HTTPS になるかが決まります。 **重要:** パッケージをサブスクライバに送信するときにパブリッシャーに SSL を使用させたい場合は、ここで HTTPS ポートを指定します。
[ローカルパスワード]	ローカルユーザ名のパスワード。

フィールド	説明
[通知先電子メールアドレス]	パブリッシャーサーバがパッケージをリリースするとき、またはパッケージが配信されるときに通知する電子メールアドレス。
[自動プル]	新規リリースが使用可能になったときにサブスクライバサーバがパブリッシャーからパッケージを自動的にプルするかどうかを指定します。 [**はい**] を選択するときは、パブリッシャーサーバからサービス呼び出し電子メールが送信される電子メールサーバ上のユーザの電子メールアドレスも指定する必要があります。 サブスクライバサーバは、電子メールポートを介して、サービス呼び出し電子メールのこのアドレスを定期的にチェックします。サブスクライバサーバは電子メールを処理するとパッケージをプルします。 サービス呼び出し電子メールには、パッケージをサブスクライバサーバの Inbound ディレクトリにロードするサービス (サブスクライバサーバ上で実行する) の呼び出し機能があります。 自動プルを使用するには、電子メールポートを自動プルアドレスで受信待機するよう設定する必要があります (以下に説明)。 電子メールポートの設定については、203 ページの「電子メールポートの追加」を参照してください。
[自動プル用電子メールアドレス]	パッケージの新規リリースが使用可能になったときにパブリッシャーサーバからサービス呼び出し電子メールが送信される電子メールアドレス。 通知先電子メールアドレスとサービス呼び出し電子メールのアドレスは異なります。たとえば、通知先電子メールは `package_notifications@mymailserver.com` に送信し、サービス呼び出し電子メールは `package_autopulls@mymailserver.com` に送信します。 自動プルを使用するには、電子メールポートをこのアドレスで受信待機するよう設定する必要があります。 電子メールポートの設定については、203 ページの「電子メールポートの追加」を参照してください。

6. [**サブスクリプションの開始**] をクリックします。

重要:
指定したサーバ上に存在しないパッケージのサブスクリプションを要求した場合、またはそのサーバがそのパッケージを所有していない場合 (当サーバがパッケージのサブスクライバの場合) は、エラーメッセージが返され、パブリッシャーサーバは要求されたサブスクリプションを処理しません。

サブスクリプション情報の更新

サブスクリプション (サブスクライバサーバ上のユーザ名、パスワードなど) に関する情報を更新するには、以下の手順に従います。

サブスクリプション情報を更新するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [サブスクライブ] に移動します。
3. [**リモートパッケージの更新とアンサブスクライブ**] をクリックします。
4. 更新するパッケージの [**更新**] 列で [**編集**] をクリックします。
5. サブスクリプション情報を変更するために、下記のフィールドに必要な情報を入力します。

フィールド	説明
[パッケージ]	サブスクリプション情報を変更するパッケージ。 新規パッケージのパブリッシュとサブスクライブをまだ実行していない場合は、パッケージを別のパッケージに変更できます。これは、同じパッケージのサブスクライブとパブリッシュを同時に実行できないからです。
[パブリッシャーのエイリアス]	パブリッシャーに割り当てるエイリアス。サブスクライバは、サブスクリプション登録のために必要なパブリッシャーサーバとの接続方法をエイリアスの定義から知ることができます。エイリアスには、ホスト名または IP アドレスなどの接続情報が含まれています。このパブリッシャーのエイリアスを定義していない場合は、[リモートサーバ] ページへのリンクをクリックします。このページで、パブリッシャーのエイリアスを設定できます。詳細については、133ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。
[ローカルポート]	サブスクライバがパブリッシャーから送信されるパッケージを受け取るための、受信待機ポートの番号。この設定によって、パブリッシャーの送信手段が HTTP になるか HTTPS になるかが決まります。 **重要:** パッケージをサブスクライバに送信するときにパブリッシャーに SSL を使用させたい場合は、ここで HTTPS ポートを指定します。 **注:** パブリッシャーは、サブスクライバに接続すると、そのデフォルトの認証 ([セキュリティの設定] ページで指定された) を使用します。ここで指定するポートがその認証を受け付けるようにします。
[ローカルユーザ名]	パブリッシャーがサブスクライバにユーザとしてログインするときのユーザ名。

フィールド	説明
	このユーザは、Replicators ACL に割り当てられるユーザグループに属している必要があります。このユーザを削除するか、またはそのユーザと Replicators ACL との関連付けを変更すると、パブリッシャーはこのパッケージをサブスクライバに送信できなくなります。
[ローカルパスワード]	ローカルユーザ名のパスワード。
[通知先電子メールアドレス]	パブリッシャーサーバがパッケージをリリースするとき、またはパッケージが配信されるときに通知する電子メールアドレス。
[自動プル]	新規リリースが使用可能になったときにサブスクライバサーバがパブリッシャーからパッケージを自動的にプルするかどうかを指定します。 自動プルを既に設定しており、無効にする場合、[**いいえ**] を選択します。次に、[**自動プル用電子メールアドレス**] フィールドに移動し、電子メールアドレスを削除します。 サーバを自動プルに設定する場合は、[**はい**] を選択します。この場合には、パブリッシャーサーバからサービス呼び出し電子メールが送信される電子メールサーバ上のユーザの電子メールアドレスも指定する必要があります。 サブスクライバサーバは、電子メールポートを介して、サービス呼び出し電子メールのこのアドレスを定期的にチェックします。サブスクライバサーバは電子メールを処理するとパッケージをプルします。 サービス呼び出し電子メールには、パッケージをサブスクライバサーバの Inbound ディレクトリにロードするサービス (サブスクライバサーバ上で実行する) の呼び出し機能があります。 自動プルを使用するには、電子メールポートを自動プルアドレスで受信待機するよう設定する必要があります (以下に説明)。 電子メールポートの設定については、203 ページの「電子メールポートの追加」を参照してください。
[自動プル用電子メールアドレス]	パッケージの新規リリースが使用可能になったときにパブリッシャーサーバからサービス呼び出し電子メールが送信される電子メールアドレス。 通知先電子メールアドレスとサービス呼び出し電子メールのアドレスは異なります。たとえば、通知先電子メールは `package_notifications@mymailserver.com` に送信し、サービス呼び出し電子メールは `package_autopulls@mymailserver.com` に送信します。 自動プルを使用するには、電子メールポートをこのアドレスで受信待機するよう設定する必要があります。

フィールド	説明
	電子メールポートの設定については、203ページの「電子メールポートの追加」を参照してください。

注:
サブスクリプションを追加するときは、パブリッシャーサーバが稼動している必要があります。

6. [**変更内容の保存**] をクリックします。

   サーバは、サブスクライバサーバとパブリッシャーサーバの両方の情報を更新します。

サブスクライブのキャンセル

サブスクリプションをキャンセルすると、サーバはパブリッシャーサーバに対してキャンセル通知を送信します。パブリッシャーサーバは、指定のパッケージのサブスクリプションリストからその要求元のサーバを削除します。サブスクリプションをキャンセルするときにパブリッシャーが稼動していなければ、次回サーバにパッケージを送信しようとしたときに、そのことがパブリッシャーに通知され、そのサブスクリプションがサブスクライバのリストから自動的に削除されます。

注:
サブスクライバがパブリッシャーによって開始されたサブスクリプションを削除すると、サブスクライバサーバはサブスクリプションリストからそのサブスクリプションを削除しますが、そのサブスクリプションがパブリッシャーのリストからすぐに削除されるわけではありません。次回、パブリッシャーサーバがサブスクライバにパッケージを送信しようとすると、パブリッシャーにその削除が通知され、パブリッシャーのリストからサブスクリプションが削除されます。

別のサーバ上のパッケージに対するサブスクリプションをキャンセルするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [サブスクライブ] に移動します。
3. [**リモートパッケージの更新とアンサブスクライブ**] をクリックします。
4. サブスクリプションをキャンセルするパッケージを検索して、✕アイコンをクリックします。

別のサーバからパブリッシュされたパッケージのインストールについて

別のサーバが使用中のサーバにパッケージをパブリッシュした場合、そのパッケージをインストールする必要があります。パッケージに対して、次のタスクを実行することもできます。

- **パッケージを回復する**サーバ上の既存のパッケージと同じ名前のパッケージをインストールすると、サーバは新しいパッケージをインストールする前に元のパッケージを *Integration Server_directory* \instances*instance_name*\replicate\salvage にコピーします。このため、新しいパッケージに満足できないときは旧バージョンを簡単に復旧することができます。前バージョンの復旧については、715 ページの「パッケージの回復」を参照してください。
- **パッケージをアクティブにする**パッケージがインストールされたらすぐにそれをアクティブにするかどうかは選択可能です。Integration Server がインストール時にパッケージをアクティブにしない場合でも、Integration Server はそのパッケージを packages ディレクトリにコピーしますが、クライアントはそれを使用できなくなります。そのパッケージをクライアント側で使用できるようにするには、それを手動でアクティブにします。詳細については、712 ページの「パッケージのアクティブ化」を参照してください。
- **パッケージをアーカイブする**パッケージのインストール後に、パッケージを自動的にアーカイブすることもできます。パッケージを自動的にアーカイブする場合、Integration Server はパッケージを *Integration Server_directory* \instances*instance_name*\replicate\inbound ディレクトリから *Integration Server_directory* \instances*instance_name*\replicate\archive ディレクトリに移動します。インストール後にパッケージを自動的にアーカイブしない場合でも、後からアーカイブすることができます。詳細については、716 ページの「パッケージのアーカイブ」を参照してください。

別のサーバからパブリッシュされたパッケージのインストール

別のサーバからパブリッシュされたパッケージをインストールするときは、以下の点に留意してください。

- パブリッシャーから送信されるパッケージには、既にそれと関連付けられたリスト ACL、特に、パブリッシャーサーバ上のパッケージに割り当てられたリスト ACL が存在します。

  インストールユーザはその ACL のメンバーではなくてもパッケージをインストールできますが、サブスクライバサーバ上のユーザは、当該パッケージのリスト ACL のメンバーでないとパッケージを Integration Server Administrator 上のページに表示することはできません。

  重要:
  パッケージをインストールするサーバに、パッケージのリスト ACL と同じ名前の ACL があることを確認してください。リスト ACL が存在しない場合は、ユーザが Integration Server Administrator のページでパッケージを表示することはできません。

- インストールするパッケージがサーバに存在しない別のパッケージに対する依存性を持つ場合、サーバはそのパッケージをインストールしますが有効にはしません。依存性が満たされるまで、インストールされたパッケージを有効にすることはできません。
- インストールするパッケージが電子メールリスナーと関連付けられる場合、サーバはそのパッケージをインストールしますがリスナーを有効にはしません。これは、Integration Server が電子メールサーバに接続するために必要なパスワードが、リスナーに関するその他の設定情報と共に設定されていないからです。リスナーを有効にするには、[**サーバ**] > [ポート] > [電子メールクライアント設定の編集] に移動し、[**パスワード**] フィールドを更新して、電子メールサーバへの接続に必要なパスワードを指定します。

- 別の Integration Server からパッケージをインストールする場合、新しいパッケージに関連付けられる HTTP URL エイリアスは、Integration Server で既に定義されている HTTP URL エイリアスと同じ名前にすることができます。Integration Server が重複エイリアス名を検出すると、**[パッケージ] > [管理] > [*package_name*]** ページにロード警告が表示されます。重複 HTTP URL エイリアスの詳細については、459 ページの「URL エイリアスの移植性と競合の可能性」を参照してください。
- 別の Integration Server からパッケージをインストールする場合、新しいパッケージのポートのポートエイリアスが、Integration Server で既に定義されているポートのポートエイリアスと同じ名前になることがあります。ポートエイリアスは、Integration Server 内で一意にする必要があります。Integration Server が重複ポートエイリアスを検出すると、**[パッケージ] > [管理] > [*package_name*]** ページにロード警告が表示されます。また Integration Server はポートを作成せずに、次の警告を server.log に書き込みます。

  [ISS.0070.0030W] ポート portNumber 上で protocol リスナーを作成中に重複エイリアス duplicateAlias が発生しました

  注:
  パッケージのロード時にポートを作成する場合、該当するエイリアスがある既存のポートを削除して、削除したポートと同じプロパティを持つ新しいポートを作成し、次に重複エイリアスがあるポートを含むパッケージを再ロードします。Integration Server は、パッケージの再ロード時にポートを作成します。
- バージョン 9.5 よりも前の Integration Server からリリースされたパケージをインストールすると、パッケージに関連付けられた各ポートに対して、Integration Server によりエイリアスが作成されます。Integration Server によって作成されるポートエイリアスの詳細については、170 ページの「ポートエイリアスについて」を参照してください。
- パッケージのホット展開が有効になっている場合、Integration Server はホット展開プロセスを使用してパッケージをインストールします。ホット展開については、742 ページの「パッケージのホット展開」を参照してください。

重要:
インストールするパッケージが正当なソースから送信されたもの (別のサーバから送信された複製のパッケージなど) かどうかを確認してください。正当かどうかが判断できない場合は、パッケージを更新した人が有資格者であったことを、同組織内の開発者と確認してください。不明なパッケージは、サーバに損傷を与えるコードを含む場合があります。

別のサーバからパブリッシュされたパッケージをインストールするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**受信リリースのインストール**] をクリックします。
4. [**リリースファイル名**] ドロップダウンリストから、インストールするパッケージを選択します。

5. インストール後直ちに Integration Server でパッケージを使用可能にする場合は、[**オプション**] フィールドの [**インストールと同時にアクティブにする**] チェックボックスをオンにします。

6. インストール後に Integration Server でパッケージを自動的にアーカイブする場合は、[**インストールと同時にアーカイブする**] チェックボックスをオンにします。このチェックボックスは、デフォルトでオンになっています。

7. [**リリースのインストール**] をクリックします。

   Integration Server によってパッケージがインストールされ、そのことを示すメッセージが表示されます。

パッケージクラスローダの使用

デフォルトでは、パッケージ内のサービスが記憶領域にクラスをロードする必要がある場合、パッケージクラスローダはクラスの検索を親クラスローダに任せます。親クラスローダは、常に Integration Server クラスローダです。ただし、Integration Server でパッケージクラスローダを代わりに使用する場合もあります。たとえば、Integration Server クラスローダがアクセスするバージョンではなく、パッケージディレクトリに存在する新しいバージョンのクラスを使用する場合などです。

パッケージクラスローダが、ロードするクラスにアクセスできる場合、そのクラスを含む jar ファイルが次のディレクトリに存在している必要があります。

Integration Server_directory \instances*instance_name*\packages*packageName*\code\jars ディレクトリ。

パッケージクラスローダを使用するように Integration Server を設定するには

1. *Integration Server_directory* \instances*instance_name*\packages*packageName* ディレクトリに移動します。この *packageName* は、ロードする jar ファイルを含むパッケージの名前です。

2. manifest.v3 ファイルに次のプロパティを追加します。

```
<value name='classloader'>package</value>
```

3. パッケージを再ロードします。

   Integration Server がクラスをロードする方法の詳細については、以下を参照してください。
   963 ページの「Integration Server におけるクラスロード 」

パッケージのホット展開

パッケージのホット展開とは、著しいダウンタイムなしに Integration Server アセットを処理できるようにしながら、実稼働環境内の Integration Server でカスタムパッケージをシームレスにインストールおよびアップグレードするプロセスのことです。Integration Server では、Integration

Server Administrator および Deployer を使用したパッケージのホット展開をサポートしています。

パッケージの新しいバージョンを展開すると、Integration Server はメモリから既存のパッケージ情報をアンロードして、パッケージとその内容の新規バージョンをメモリにロードします。パッケージのアップグレード中、一部のアセットが一定時間利用できなくなることがあります。そのため、サービス拒否、実行中タスクの中断など、重大な問題が発生する可能性があります。パッケージのホット展開では、パッケージは常に使用可能な状態が保証されます。つまり、ダウンタイムはありません。これは実稼働環境では重要です。

従来、Integration Server クラスタ全体をシャットダウンせずにクラスタ内でパッケージをアップグレードするために、ローリング休止およびアップグレードプロセスが使用されてきました。クラスタのローリングアップグレードでは、クラスタ内の各 Integration Server を個別に休止し、アップグレードして再起動している間、クラスタ内の他のサーバは要求を処理し続けます。このアプローチには、Integration Server が休止状態のときにはすべての外部ポートが無効化されるため、Integration Server が要求を処理したりサービスを実行したりできる状態にない、という制限があります。パッケージのホット展開を使用すると、各 Integration Server を休止せずにパッケージを簡単に展開できるようになります。ホット展開の場合は、展開されるパッケージに含まれるアセットまたはそのパッケージに依存するアセットのみが影響を受け、サーバ全体に影響はありません。ただし、修正のインストールやその他保守タスクなどを実行するときは、Integration Server の休止モードが不可欠です。

ホット展開の仕組み

パッケージのホット展開を実行するように Integration Server を設定すると、Integration Server ではブロックして待機するアプローチが採用されます。つまり、Integration Server は展開しているパッケージに関連する実行中タスクが完了するまで待機してから、既存パッケージの情報をメモリからアンロードします。同時に、パッケージのホット展開を開始すると、Integration Server は展開しているパッケージに属するアセットやそのパッケージに依存しているアセットへのアクセスをブロックします。

Integration Server では、以下の手順に従うことにより、アセットの処理で失敗がないようにします。

- 展開しているパッケージに依存するすべての実行中タスクが完了するまで待機します。Integration Server がパッケージの展開を試みる前に実行中タスクの完了を待機しなければならない時間を指定できます。指定されたタイムアウト期間にタスクが完了しない場合、ホット展開は失敗します。Integration Server では、ダウンタイムなしを保証するため、ホット展開が失敗したときに元のパッケージを回復することもできます。
- 展開しているパッケージに属するアセットやそのパッケージに依存しているアセットの実行をブロックします。たとえば、呼び出しているサービスが展開対象のパッケージに属する場合、パッケージのアップグレードが終わるまで Integration Server はサービスの呼び出しをブロックします。どのアセットに依存性があるか判断する場合、Integration Server ではパッケージの依存性を明示的に指定したアセットだけが考慮されます。

ホット展開時のパッケージの依存性の判断

実行中タスクが中断されないようにするため、パッケージの依存性を正しく設定することが重要です。パッケージのホット展開時に、Integration Server では展開しているパッケージのすべて

のアセットと識別されたそのすべての依存関係について、ブロックして待機するアプローチを採用します。依存性を設定していない場合、Integration Server ではアセットの呼び出しを行う可能性があります。その時点でアセットがアップグレード中である場合、エラーが発生し、呼び出しが失敗することがあります。

たとえば、「FinanceUtil」が「Finance」パッケージの依存パッケージであると明示的に指定しなかった場合、Integration Server は「FinanceUtil」パッケージ内のアセットの実行が完了するまで待機せずに、「Finance」パッケージを展開します。そのため、「Finance」パッケージのホット展開は失敗する可能性があります。Integration Server が「FinanceUtil」パッケージ内のアセットの実行が完了するまで待機してから「Finance」パッケージをホット展開するのは、「FinanceUtil」が「Finance」パッケージの依存パッケージであると明示的に指定した場合に限られます。パッケージの依存性の詳細については、『*webMethods Service Development Help*』を参照してください。

パッケージのホット展開に関する考慮事項

ホット展開を設定または実行する前に、以下の動作、制限および考慮事項に留意してください。

- ホット展開を有効にすると、Integration Server は、新しいパッケージかアップグレードされたパッケージかに関わらず、任意のカスタムパッケージのインストールのためにホット展開を実行します。
- パッケージ全体のホット展開のみ実行できます。パッケージ内の特定アセットに対してホット展開を実行できません。ホット展開中、Integration Server はパッケージのすべてのアセットをアップグレードします。
- パッケージのホット展開を実行するときに、Integration Server ノードを休止または再起動する必要はありません。
- ホット展開は Integration Server によって所有されたアセットにのみサポートされています。たとえば、フローサービスと Java サービスです。
- Integration Server では、パッケージのホット展開時に、HTTP/HTTPS/FTP/FTPS 要求、スケジューラタスク、ファイルポーリング要求、電子メール、メッセージング/JMS 要求、Web サービス要求などのプロセスが続行するようにします。展開中、Integration Server が展開しているパッケージに属するアセットやそのパッケージに依存しているアセットの実行をブロックするため、パッケージのホット展開の影響を受けるアセットからの応答で遅延が発生する可能性があります。

  たとえば、ホット展開中、Integration Server が既存のパッケージ情報をメモリからアンロードした場合、Integration Server は展開されたパッケージに関連付けられたポートを無効にします。このようなポートは、パッケージのアクティブ化が成功すると有効化されます。したがって、ポートが関連付けられたパッケージをホット展開する場合、ポートの有効化が成功するまで、このようなポート経由のサービスまたは他のリソースへのアクセスは失敗します。

  ホット展開されているパッケージのポートが無効になる状況を避けるには、ホット展開で展開されないパッケージにポートを割り当てます。

  注:

ポートを作成する場合、ポートをパッケージに割り当てます。Integration Server では、ポートのリスナー設定情報は、指定したパッケージに保存されます。ポートを含むパッケージの有効化または無効化によって、ポートも有効化または無効化されます。

- 展開しているパッケージまたは依存パッケージ内のアセットが展開の時点で処理中の場合、Integration Server は処理の完了、または指定されたタイムアウト期間のいずれか早いほうが経過するまで待機してから、パッケージの展開を試行します。指定されたタイムアウト期間にタスクが完了しない場合、ホット展開は失敗することがあります。ホット展開を確実に成功させるには、ホット展開のタイムアウト値をタスクの完了にかかる時間よりも大きい値に設定する必要があります。

  スケジュール済みユーザタスクに関して、Integration Server は展開対象パッケージの依存性を評価し、すでに実行中のスケジュール済みユーザタスクの完了を許可します。ホット展開が完了するまで、Integration Server はまだ開始されていないスケジュール済みユーザタスクを一時停止します。

- ホット展開はクラスタに対応していません。パッケージのホット展開は、クラスタ内のすべてのサーバノードで同時に発生しないことがあります。また、1 つのサーバノードにおけるホット展開に関連する情報メッセージやデバッグメッセージは、クラスタノード間で共有されません。

- パッケージの新しいバージョンの展開が失敗した場合に備えて、Integration Server には、Integration Server がパッケージの以前のバージョンを自動的に回復できるようにする自動設定オプションが用意されています。

  注:
  Integration Server でパッケージを初めてインストールするときに、ホット展開でエラーが発生すると、Integration Server はパッケージを自動的に回復できません。これは、パッケージの作業コピーが存在しないためです。

- ホット展開操作をキャンセルするために pub.packages.hotdeployment:cancel サービスを使用できます。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

カスタムパッケージのホット展開の有効化と設定

ホット展開を有効にすると、Integration Server は、新しいカスタムパッケージまたはアップグレードされたバージョンのカスタムパッケージをインストールする際にホット展開を実行します。Integration Server のホット展開を設定する前に、743 ページの「ホット展開時のパッケージの依存性の判断」および744 ページの「パッケージのホット展開に関する考慮事項」の情報を確認してください。

カスタムパッケージのホット展開を有効化および設定するには

1. Integration Server Administrator で、**[設定] > [ホット展開] > [ホット展開設定の編集]** をクリックします。

2. [**有効**] に [**はい**] を選択してパッケージのホット展開を有効にします。

3. [**タイムアウト**] フィールドで、Integration Server がパッケージのホット展開前に実行中の処理の完了を待機する最大秒数を指定します。実行に時間のかかるタスクをパッケージの展開前に完了させる場合、タイムアウト値はタスクの完了にかかる時間よりも大きい値に設定します。デフォルトは 60 秒です。

4. パッケージの新しいバージョンの展開に失敗したときに Integration Server によってパッケージの以前のバージョンに復帰させる場合は、[**自動回復**] オプションで [**はい**] を選択します。[**自動回復**] オプションで [**はい**] を選択すると、Integration Server は新しいバージョンを展開する前に、元のパッケージを *Integration Server_directory* \instances*instance_name*\replicate\salvage ディレクトリにコピーします。そのため、ホット展開時にエラーが発生した場合は、Integration Server によって回復され、元のパッケージに復帰します。

5. [**変更内容の保存**] をクリックします。

カスタムパッケージのホット展開の無効化

カスタムパッケージのインストールのホット展開を無効にするには、以下の手順に従います。

カスタムパッケージのホット展開を無効にするには

1. Integration Server Administrator で、**[設定] > [ホット展開] > [ホット展開設定の編集]** をクリックします。

2. [**有効**] に [**いいえ**] を選択してパッケージのホット展開を無効にします。

3. [**変更内容の保存**] をクリックします。

パッケージの自動展開

パッケージの自動展開により、Microservices Runtime はパッケージを自動的にインストールまたはアップグレードできます。パッケージの自動展開が有効になっている場合、Deployer または Microservices Runtime Administrator などの管理者ツールを使用せずに、パッケージをインストールまたはアップグレードできます。パッケージの自動展開はオンプレミスの Microservices Runtime および Docker コンテナの内部で実行されている Microservices Runtime で使用できます。

パッケージの自動展開は、Docker コンテナで実行されている Microservices Runtime で特に役に立ちます。コンテナにインストールまたは更新するパッケージは、Microservices Runtime が定期的にスキャンする `autodeploy` ディレクトリに入れることができます。Microservices Runtime は新規パッケージまたは更新したパッケージを検索しインストールします。自動展開を使用しない場合、Docker イメージを再構築して新規または更新されたパッケージを Microservices Runtime コンテナで使用する必要があります。詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

注:

パッケージの自動展開機能は、webMethods Microservices Runtime ではデフォルトで使用可能です。Integration Server でパッケージの自動展開機能を使用するには、Integration Server に追加ライセンスが必要です。

38 サービスの管理

- サービスとは 750
- サービスの完全修飾名 750
- サービスとフォルダに関する情報の検索 752
- 手動によるサーバへのサービスの追加 754
- サービスのテスト 754
- サービスに関連付けられたスレッドのキャンセルと強制終了 754
- パッケージのロード、アンロードまたは複製時に実行するサービス 757
- 特定のイベントに応答するサービスの実行 758
- グローバル変数の管理 759

サービスとは

サービスは、クライアントが呼び出すことのできる機能ユニットで、サーバ上に常駐しています。サービスは、1 つのアプリケーション全体である場合もあれば、大規模なアプリケーションの一部として使用される場合もあります。サービスには、フローサービス (Web サービスコネクタなど)、アダプタサービス、Java サービス、C/C++ サービスなど、いくつかのタイプがあります。

Software AG Designer を使用すると、すべてのフローサービスを作成できます。なお、データベースフローサービスは、Integration Server Administrator でも作成することができます。また、Designer を使用してアダプタサービスや Java サービスを作成したり、独自の開発環境を使用して Java サービスおよび C/C++ サービスを作成したりすることもできます。サービスのタイプの詳細およびサービスの作成方法については、『*webMethods Service Development Help*』を参照してください。

パッケージ内の 1 つ以上のサービスを開始サービス、シャットダウンサービスまたは複製サービスとして指定できます。開始サービスは、パッケージのロード時にサーバで自動的に実行されるサービスです。シャットダウンサービスは、パッケージのアンロード時にサーバで自動的に実行されるサービスです。複製サービスは、パッケージの複製時にサーバで自動的に実行されるサービスです。

サービスのパフォーマンスを向上させるには、サーバがサービス結果をキャッシュに保存するように設定します。サービス結果がキャッシュに保存されると、次回に同じサービスに対する要求を受信したときには、サーバはサービスを実行する代わりに、キャッシュされている結果を返します。詳細については、783 ページの「サービス結果のキャッシング」を参照してください。

Integration Server のスケジュール機能を使用して、指定したときにサービスが実行されるようにスケジュールできます。サービスのスケジュールの詳細については、763 ページの「サービスのスケジュール」を参照してください。

サービスの完全修飾名

サービスの完全修飾名は、「フォルダ識別子」と「サービス名」の 2 つの部分で構成されます。フォルダ識別子には、1 つ以上のフォルダ名が含まれています。サービス名はサービスの単一の名前です。

関連サービスをグループ化するには、フォルダを使用します。フォルダに他のフォルダが含まれる場合、ネストされているフォルダはサブフォルダと呼ばれます。たとえば、財務情報に関連する複数のサービスがある場合、「Finance」という名前のフォルダを作成して、財務関連サービスを入れます。また、財務サービス内に個人の財務に関するサービスが含まれているとします。この場合、「Personal」という名前のサブフォルダを作成して、個人の財務に関するサービスを入れます。

サービス名には任意の名前を使用できます。たとえば、財務サービスの 1 つに株価情報が含まれている場合、「StockQuote」というサービス名を付けます。

サービスの完全修飾名を指定するには、次の形式のように、フォルダ名、コロン (:)、サービス名の順に記述します。

folder:service

たとえば、「StockQuote」サービスが「Finance」フォルダにある場合、サービスの完全修飾名は次のようになります。

Finance:StockQuote

フォルダ名の部分に 2 つ以上のフォルダがある場合は、次のように各フォルダ名をピリオドで区切ります。

folder.subfolder1.subfolder2:service

たとえば、「HomeLoan」サービスが「Personal」フォルダにあるとします。Personal フォルダは「Finance」フォルダに含まれているので、サービスの完全修飾名は次のようになります。

Finance.Personal:HomeLoan

各サービスの完全修飾名は、必ずそのサーバ内で一意の名前になるようにしてください。また、サービスの完全修飾名は、サーバに常駐するどの仕様やドキュメントタイプの完全修飾名とも重複しないようにしてください。

注:
IntegrationServer_directory\instances*instance_name*\config ディレクトリにある server.cnf ファイル内の `watt.server.illegalNSChars` 設定に、フォルダとサービスの名前に使用できない文字が定義されています。使用できない文字の設定を表示または変更するには、146 ページの「拡張設定の使い方」の説明に従って、Integration Server Administrator で [設定] > [拡張] ページを使用します。

パッケージ名およびサービス名

パッケージ名とフォルダ名との関係によって、混乱が生じる場合があります。サービスが属しているパッケージの名前は、含まれているサービスやフォルダの名前に影響を及ぼすものではありません。また、パッケージ名がクライアントアプリケーションからのサービスの参照に影響を及ぼすこともありません。たとえば、「Admin」というパッケージにある「Personnel:GetDeptNames」というサービスを「EmployeeData」というパッケージに移動した場合でも、このサービスを参照するクライアントアプリケーションに影響が及ぶことはなく、「Personnel:GetDeptNames」というサービス名のままで参照できます。

各サービスの完全修飾名はサーバ内で一意の名前である必要があるため、同じサーバにある 2 つの異なるパッケージ内のサービスに同一の名前を付けることはできません。

サービスの HTTP URL エイリアス

サービスを呼び出すためにユーザが指定する HTTP URL のエイリアスを作成できます。エイリアスの方が、完全な URL パス名よりも入力しやすく、Integration Server でリソースを移動するのが簡単です。また、ユーザにディレクトリ名とファイル名が表示される URL パス名より安全です。

HTTP URL エイリアスは、Designer または Integration Server で作成できます。詳細については、449 ページの「HTTP URL エイリアスの設定」を参照してください。

サービスとフォルダに関する情報の検索

ここでは、サーバにあるサービスやフォルダの一覧を参照して、特定のサービスに関する情報を表示する方法について説明します。

フォルダとサービスの一覧

Integration Server Administrator の [**フォルダとサービス**] ページには、サーバに常駐しているサービスと、そのサービスが関連付けられているフォルダが一覧表示されます。

フォルダとサービスを一覧表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**フォルダの表示**] をクリックします。
4. フォルダの内容を表示するには、フォルダ名をクリックします。別の [**フォルダの一覧**] ページが表示されます。選択したフォルダについて、サブフォルダ、サービスの順に表示されます。

   選択したフォルダ内のサブフォルダとサービスを表示するには、引き続きフォルダ名を順にクリックします。

   注:
   リストアクセスが付与されているフォルダとエレメントのみが表示されます。

サービスに関する情報の表示

[パッケージ] > [管理] > [WmPublic] > [サービス] > ***[service]*** ページには、選択したサービスや仕様に関するさまざまな情報が表示されます。

サービスに関する情報を表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**パッケージの一覧**] で、表示するサービスが属しているパッケージをクリックします。
4. [*packagename* **内のサービスを表示**] をクリックします。

5. 情報を表示するサービスまたは仕様の名前をクリックします。サービスに関して Integration Server Administrator が表示する情報の詳細については、753 ページの「サービス情報」を参照してください。

サービス情報

Integration Server Administrator では、サービスに関して次の情報が表示されます。

セクション名	説明
[一般情報]	次の識別情報が表示されます。 ■ サービスが含まれているフォルダとサービスの名前 ■ サービスが関連付けられているパッケージの名前 ■ サービスのタイプ (フロー、Java、C/C++) ■ サービスがステートレスかどうか ■ 許可されている HTTP メソッド
[ユニバーサル名]	サービス名の修飾に使用される名前。ネームスペース名とローカル名で構成されています。 慣例では、一般的に URI をネームスペース名として使用します (たとえば、http://www.gsx.com/gl)。URI を使用することで、確実にユニバーサル名をグローバルに一意の名前とすることができます。 ローカル名は、ネームスペース名に含まれるコレクション内でサービスを一意に識別する名前です。ほとんどのサイトでは、修飾されていないサービス名をローカル名として使用しています。 ネームスペース名とローカル名には、文字または数字を任意の順序で使用できます。
[Java 用のパラメータ]	Java サービス用の識別情報 (サービスの Java クラス名とメソッド名)。
[アクセスコントロール]	サービスまたは仕様に割り当てられている ACL。ACL、サービスおよび仕様の詳細については、534 ページの「ACL によるリソースへのアクセス制御」を参照してください。
[キャッシュの制御]	サービスの実行結果をサーバのキャッシュに保存するかどうかの設定。詳細については、783 ページの「サービス結果のキャッシング」を参照してください。
[データ形式]	バインディングサービスの名前。サービスの受信 XML データ、サービスに関連付けられている出力テンプレート (存在する場合)、および出力テンプレートのタイプ (HTML または XML) を解析するサービスです。出力テンプレートの詳細については、『Dynamic Server Pages and Output Templates Developer's Guide』を参照してください。

手動によるサーバへのサービスの追加

Designer または Developer を使用して作成されなかった Java サービスまたは C/C++ サービスがある場合は、jcode ユーティリティを使用して、手動でサービスをサーバに追加する必要があります。詳細については、『*webMethods Service Development Help*』のユーザ独自の IDE を使用した Java サービスの構築に関する情報を参照してください。

サービスのテスト

Integration Server Administrator を使用して、サービスの動作をテストすることができます。サービスの動作を迅速かつ容易に検証し、例外の入力値を使用してテストすることができます。

注:
Integration Server Administrator は限定的テスト環境を提供し、タイプストリングの変数の入力値のみ指定できます。Designer によって、よりパワフルなサービスのテスト環境が提供されます。

サービスをテストするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**パッケージの一覧**] から、テストするサービスが属しているパッケージをクリックします。
4. [***packagename* 内のサービスを表示**] をクリックします。
5. テストするサービスの名前をクリックします。
6. サービスをテストするには、[***servicename* のテスト**] をクリックします。

   [***ServiceName* のテスト**] ページが表示されます。
7. 入力値を使用してサービスをテストする場合、ページの [**入力値の割り当て**] セクションに必要な入力情報を入力して、[**テスト (入力値あり)**] をクリックします。

   特定の入力値を使用せずにサービスをテストする場合は、[**テスト (入力値なし)**] をクリックします。

サービスに関連付けられたスレッドのキャンセルと強制終了

フローサービスまたは Java サービスが外部のリソースを待機しているか、または無限ループになっているため、応答がなくなっていると思われる場合は、サービスに関連付けられた 1 つ以上のスレッドの実行を停止できます。次の 2 つのオプションがあります。

- **スレッドをキャンセルする**スレッドをキャンセルした場合、Integration Server はそのスレッドで保持されているリソースを解放します。たとえば、スレッドが JDBC 接続を保持している場合、Integration Server はこの接続を閉じて解放し、JDBC 接続プールに返します。
- **スレッドを強制終了する**スレッドを強制終了すると、Integration Server はリソースの解放を試みます。リソースを解放できない場合は、継続してスレッドを終了しようとします。Integration Server は、プールに新しいスレッドを補充します。

Integration Server では、スレッドを強制終了する前に、スレッドのキャンセルを試みる必要があります。スレッドのキャンセルに成功した場合は、スレッドを強制終了する必要はありません。

スレッドをキャンセルまたは強制終了すると、Integration Server は、実行中の機能に応じて、グローバル、トリガー、スケジューラまたはサーキットブレーカースレッドプールにスレッドを返します。

Integration Server は、[**システムスレッド**] ページでキャンセルまたは強制終了できるスレッドにフラグを設定して、これらのスレッドを次のように表示します。

- キャンセルできるスレッドは、[キャンセル] 列に ✓ とマークされています。
- 強制終了できるスレッドは、[強制終了] 列に ✕ とマークされています。

サービススレッドをキャンセルまたは強制終了する機能は、watt.server.threadKill.enabled パラメータによって制御されます。スレッドをキャンセルまたは強制終了できるようにする場合は、このパラメータを「true」に設定する必要があります (デフォルト設定)。

注:
キャンセルされたスレッドを使用するサービスを Integration Server で中断する場合、watt.server.threadKill.interruptThread.enabled パラメータを「true」に設定する必要があります。

スレッドのキャンセルまたは強制終了

スレッドのキャンセルまたは強制終了を行う前に、以下の情報に留意してください。

- スレッドをキャンセルまたは強制終了できるようにする場合は、watt.server.threadKill.enabled パラメータを「true」に設定する必要があります (デフォルト設定)。
- Integration Server では、スレッドを強制終了する前に、スレッドのキャンセルを試みる必要があります。
- JDBC 接続がデータベースからの応答を待機している場合、スレッドをキャンセルまたは強制終了できません。
- 基盤となるサービスがブロッキング IO を待機している場合、スレッドをキャンセルまたは強制終了できません。ブロッキング IO の例を以下に示します。
  - HTTP 要求からの応答を待機するサービス
  - FTP 要求からの応答を待機するサービス

  - ファイルまたはソケットでの読み取り/書き込み要求からのデータの受信を待機するサービス
- ユーザが作成した Java サービスまたはフローサービスのスレッドのみキャンセルまたは強制終了が可能です。
- Integration Server システムサービスのスレッドをキャンセルまたは強制終了することはできません。

注意:
スレッドのキャンセルまたは強制終了は、注意して行ってください。スレッドをキャンセルしても、サービスが保持しているリソースが解放されない場合があります。スレッドを強制終了すると、リソースが不安定な状態になる場合があります。

サービスに関連付けられたスレッドをキャンセルまたは強制終了するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [統計情報] に移動します。
3. [**システムスレッド**] フィールドの [**現在**] 列で、現在のスレッド数をクリックします。

   Integration Server が [**システムスレッド**] ページを表示します。
4. キャンセルできるスレッドの検索を簡単にするには、[**リストの先頭にありキャンセルできるスレッドを表示する**] チェックボックスをオンにします。

   Integration Server は表示を動的に更新して、キャンセルできるスレッド ([キャンセル] 列に ✓ アイコン) または強制終了できるスレッド ([強制終了] 列に ✕ アイコン) をページの上部に表示します。
5. スレッドに関する情報を表示するには、そのスレッドの [**名前**] 列で、スレッド名をクリックします。

   そのスレッドのダンプが表示されます。スレッドダンプで提供される情報を使用して、スレッドをキャンセルするかどうかを決定します。

   [**システムスレッド**] ページに戻るには、[**システムスレッドに戻る**] をクリックします。
6. キャンセルするスレッドの行の [キャンセル] 列の ✓ をクリックします。

   Integration Server が、操作を確認するプロンプトを表示します。

キャンセルが成功した場合	Integration Server は表示からスレッドを削除します。
キャンセルが失敗した場合	Integration Server は表示を更新して、[強制終了] 列に ✕ を表示します。 スレッドを強制終了する場合は、✕ をクリックします。

Integration Server が、操作を確認するプロンプトを表示します。

Integration Server がスレッドを強制終了できる場合は、表示からスレッドを削除します。

Integration Server がスレッドを強制終了できない場合、スレッドは表示されたまま残ります。

パッケージのロード、アンロードまたは複製時に実行するサービス

サーバがメモリからパッケージをロードまたはアンロードしたり、パッケージを複製したりするたびに、指定した一連の操作がサーバで自動的に実行されるようにするには、それぞれ、開始サービス、シャットダウンサービス、複製サービスを設定します。ここでは、開始サービス、シャットダウンサービスおよび複製サービスの概要について説明します。

これらのサービスを指定するには、Designer を使用する必要があります。手順については、『*webMethods Service Development Help*』を参照してください。

開始サービスとは

開始サービスは、パッケージのロード時にサーバが自動的に実行するサービスです。次のような場合に、サーバはパッケージをロードします。

- サーバを初期化するとき (パッケージが有効な場合)
- ユーザが Integration Server Administrator を使用してパッケージを再ロードするとき
- ユーザが Integration Server Administrator を使用してパッケージを有効にするとき

開始サービスは、初期化ファイルを生成する場合や、サーバがパッケージをロードする前に環境を評価および準備 (たとえば、セットアップやクリーンアップ) する場合に役立ちます。このような場合に限らず、どのような目的でも開始サービスを使用できます。たとえば、キャッシュされた結果をクライアントアプリケーションで直ちに使用できるように、時間のかかるサービスを起動時に実行しておくことができます。

シャットダウンサービスとは

シャットダウンサービスは、サーバがパッケージをメモリからアンロードするときに、自動的に実行するサービスです。次のような場合に、サーバはパッケージをメモリからアンロードします。

- サーバをシャットダウンまたは再起動するとき
- ユーザが Integration Server Administrator を使用してパッケージを無効にするとき
- パッケージをメモリから削除する前に、ユーザが Integration Server Administrator を使用してパッケージを再ロードするとき

シャットダウンサービスは、ファイルを閉じるときや一時データを消去するときなど、クリーンアップタスクを実行する場合に役立ちます。また、パッケージのアンロード前に、進行中の作業や状態情報をキャプチャする場合にも役立ちます。

複製サービスとは

複製サービスは、サーバがパッケージをリリースまたはアーカイブするときに自動的に実行するサービスです。複製サービスは、管理者が [パッケージ] > [パブリッシュ] > [リリースの作成と削除] ページの [リリースの作成] リンクをクリックしたとき、または、[パッケージ] > [管理] ページの [アーカイブ] アイコン をクリックしたときに実行されます。

複製サービスを使用すると、パッケージに状態情報や設定情報を残しておくことができるので、パッケージがリモートサーバでアクティブ化されたときにこれらの情報を使用することができます。

開始サービス、シャットダウンサービスおよび複製サービスの使用に関するガイドライン

パッケージの開始サービス、シャットダウンサービスおよび複製サービスを使用する場合は、次のガイドラインに留意してください。

- 開始サービスまたはシャットダウンサービスを作成する場合、これらのサービスが使用されるパッケージ内に登録してください。 複製サービスを作成する場合には、現在のパッケージを含め、サーバにロードされているパッケージ内の有効なサービスを任意に選択して登録できます。
- パッケージの開始サービスの実行が終了するまで、クライアントはパッケージ内のサービスを使用できないため、負荷の高いサーバにアクセスするサービスを開始サービスに設定することは避けてください。そうしないと、同じパッケージにある他のサービスの使用に遅延が発生します。
- 1 つのパッケージに 1 つ以上の開始サービスを割り当てることができますが、これらのサービスの実行順序は指定できません。特定の順序で実行する必要のある一連の操作がある場合、一連の操作全体を単一サービス内でエンコードするか、または 1 つの開始サービスから他の開始サービスを呼び出すように指定します。

手順については、『*webMethods Service Development Help*』を参照してください。

特定のイベントに応答するサービスの実行

Event Manager はサーバ上で稼動し、サーバの「イベント」を監視します。イベントは、Event Manager が認識し、イベントハンドラによって処理される特定のアクションです。イベントハンドラは、特定のイベントが発生したときにアクションを実行するように、開発者によって記述されているサービスです。Event Manager はさまざまな種類のイベントを認識します。たとえば、webMethods Integration Server がサーバの状態に関する例外をスローしたときに発生する「アラームイベント」があります。ユーザがサーバにログオンできない場合、ポートが開始できない場合、ユーザがポートへのアクセスを拒否された場合などに、サーバはアラームイベントを生成します。

開発者は Designer を使用して Event Manager を制御します。サーバは、イベントタイプおよびイベントサブスクリプションの情報を eventcfg.bin ファイルに保存します。eventcfg.bin ファイルは Integration Server を最初に起動したときに生成され、*IntegrationServer_directory*\instances*instance_name*\config ディレクトリに保存されます。通常は、管理者が eventcfg.bin ファイルに対して直接作業する必要はありません。ただし、Integration Server をクラスタリングする場合、使用中のサーバから別のサーバに eventcfg.bin ファイルをコピーして、クラスタにあるすべてのサーバのイベントサブスクリプションを複製する必要があります。Event Manager の使用方法の詳細については、『*webMethods Service Development Help*』を参照してください。

グローバル変数の管理

グローバル変数は、フローサービスで使用できるキーと値のペアです。設計時には、変数の値をハードコーディングせず、グローバル変数を使用するように Integration Server を設定することができます。

フローサービスでグローバル変数の置換を使用するには、パイプライン変数に値を割り当てるとき、値としてグローバル変数を指定する必要があります。また、Integration Server が変数の置換を実行するように指定することも必要です。その場合、Integration Server は実行時に、フローサービスの変数をそれぞれのグローバル変数に割り当てられた値で置換します。

たとえば、pub.client* サービスの *serverhost* または *serverport* など、入力パラメータのグローバル変数を定義できます。設計時には、パイプラインの対応する変数の値として *serverhost* または *serverport* に定義されたグローバル変数を使用するように Integration Server を設定します。その場合、Integration Server は実行時に、フローサービスの変数を *serverhost* または *serverport* グローバル変数に割り当てられた値で置換します。

グローバル変数を使用すると、パイプライン変数に割り当てられた値を変更しやすくなります。各マッピングまたはフローサービスを変更する代わりに 1 つのグローバル変数を変更して、変数に割り当てられた値を変更することができます。

グローバル変数の置換の詳細については、『*webMethods Service Development Help*』を参照してください。

グローバル変数の作成

グローバル変数は、Integration Server Administrator を使用して定義できます。Integration Server はグローバル変数定義を *Integration Server_directory* \instances*instance_name*\config\globalVariables.cnf に保存します。

グローバル変数を作成するときは、以下の点に留意してください。

- グローバル変数は、String、String List および String Table 型の変数の値を指定する場合にのみ使用できます。
- グローバル変数名
  - ピリオド (.) およびアンダースコア (_) を除き、特殊文字は使用できません。
  - 最大長は 255 文字です。

  - 大文字と小文字は区別されます。
  - Integration Server 上で一意である必要があります。
- 作成するグローバル変数がパスワードの場合、Integration Server は OPE (送信パスワード暗号化) を使用してその変数を暗号化して保存します。
- グローバル変数の最大長は 255 文字です。
- グローバル変数を編集する場合は、[**値**] フィールドのみを変更できます。
- グローバル変数は、クラスタ内のすべての Integration Server で同じとする必要があります。
- グローバル変数は、webMethods Deployer を使用して展開できます。Deployer の使用の詳細については、『*webMethods Deployer User's Guide*』を参照してください。
- サービスのデバッグ時にグローバル変数の置換を使用できます。

グローバル変数を作成するには

1. Integration Server Administrator を開きます。
2. [**設定**] > [グローバル変数] に移動します。
3. [**グローバル変数の追加**] をクリックします。
4. [**グローバル変数**] で、以下の情報を指定します。

パラメータ	指定内容
IS パスワードであるかどうか	定義するグローバル変数がパスワードであるかどうか。 [**IS パスワード**] チェックボックスをオンにした場合、[**値**] フィールドに入力するテキストはパスワードとして入力され、文字ではなく入力を示すアスタリスクが表示されます。Integration Server は、[**値**] フィールドに入力したテキストを送信パスワードとして暗号化し、保存します。
キー	グローバル変数の名前。Integration Server は、グローバル変数の置換の実行時に、このキーを使用してグローバル変数を参照します。
値	グローバル変数の値。

5. [**変更内容の保存**] をクリックします。

グローバル変数の削除

グローバル変数定義は、Integration Server Administrator を使用して Integration Server から削除できます。フローサービスが使用しているグローバル変数を、フローサービスの実行中に削除すると、Integration Server は例外をスローしてサービスの実行を停止します。

グローバル変数を削除するには

1. Integration Server Administrator を開きます。
2. [**設定**] > [グローバル変数] に移動します。
3. 削除するキーと値のペアが含まれている行を見つけて、✕アイコンをクリックします。

   グローバル変数を削除するかどうかを確認するダイアログが Integration Server で表示されます。
4. [**OK**] をクリックします。

39 サービスのスケジュール

- 概要 764
- ユーザタスクのスケジュール 764
- スケジュールされているユーザタスクの表示 769
- スケジュールされているユーザタスクの更新 771
- ユーザタスクの一時停止 771
- 一時停止中のユーザタスクの再開 773
- スケジュールされているユーザタスクのキャンセル 774
- スケジュールされているシステムタスクの表示 774
- 単純な [繰り返し] オプション 775
- [複雑な繰り返し] オプション 776
- [ターゲットノード] オプション 779
- 夏時間に対する移行がスケジュール済みタスクに及ぼす影響 781

概要

サーバのスケジュール機能を使用して、指定したときにサービスが実行されるようにスケジュールできます。スケジュールしたサービスは「ユーザタスク」と呼ばれます。 以下の操作を実行できます。

- Integration Server が実行するユーザタスクを作成する。
  - 1 回
  - 単純な間隔での繰り返し (たとえば毎日の毎時)
  - 複雑な間隔での繰り返し (たとえば 6 月の 1 週おきの火曜日)
- スケジュール済みタスクのリストを表示する。
- 既存のユーザタスクのスケジュールオプションを更新する。
- Integration Server がすべてのスケジュール済みタスクを完了する前に、スケジュール済みのユーザタスクをキャンセルする。
- タスクの実行を一時的に停止する。
- 同じデータベースに接続した、1 つ、いずれか、またはすべての Integration Server で実行するユーザタスクを作成する。
- タスクがスケジュール済みの実行時刻を過ぎても開始されていない場合に Integration Server が実行するアクションを指定する。

注:
Integration Server Administrator を使用してタスクをスケジュールするだけでなく、一連の組み込みサービスを使用してスケジュールを実行することもできます。詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

Integration Server で提供されるタスク

Integration Server には、変更可能なユーザタスクがあります。たとえば、重複抑制処理に対してドキュメント履歴データベースを設定している場合、Integration Server ではメッセージ履歴スイーパタスクが提供されます。このタスクによって、期限切れのエントリがドキュメント履歴データベースから削除されます。このタスクが Integration Server によってスケジュールされている場合でも、このサービスの実行頻度を変更することができます。

管理者がスケジュールを設定するユーザタスクのほか、通常のシステム運用において実行されるタスクを Integration Server がスケジュールする「システムタスク」と呼ばれるタスクがあります。スケジュールされているほとんどのシステムタスクを表示することはできますが、更新またはキャンセルすることはできません。

ユーザタスクのスケジュール

ユーザタスクをスケジュールするには、以下の手順に従います。

ユーザタスクをスケジュールするには、以下の手順に従います。

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [スケジューリング] に移動します。
3. [**スケジュールタスクの作成**] をクリックします。
4. 次の表に従って、[**サービス情報**] パラメータを設定します。

パラメータ	指定内容
説明	タスクの説明。
[フォルダ名.サブフォルダ名:サービス名]	Integration Server で実行するサービスの完全な名前。 [**入力の割り当て**] をクリックして、Integration Server が実行するサービスの新しい入力値を入力するか既存の値を変更します。 入力値を割り当てる前に、以下の点に注意してください。 ■ データタイプがストリングであるトップレベルの入力パラメータにのみ値を割り当てることができます。 ■ [**入力の割り当て**] を使用して入力値を割り当てると、pub.scheduler フォルダにある組み込みサービスを使用してサービスに割り当てられていた入力値が上書きされます。 サービス名の指定の詳細については、750 ページの「サービスの完全修飾名」を参照してください。
[実行ユーザ]	サービスの実行時にサーバで使用するユーザ名。 をクリックしてユーザを検索して選択します。ローカルディレクトリまたはセントラルディレクトリからユーザを選択できます。 Integration Server は、指定したユーザがサービスを呼び出した認証ユーザであると見なして、サービスを実行します。サービスが ACL で管理されている場合は、必ずサービスの呼び出しを許可されているユーザを指定してください。
[ターゲットノード]	同じデータベースに接続した他の Integration Server 上でタスクを実行するかどうか。 ■ データベースに接続した 1 つのサーバだけでタスクを実行する場合、どのサーバで実行してもかまわないときには、[**任意のサーバ**] を選択します。 ■ 設定済みクラスタ (つまり、ステートフルクラスタ) のサーバの場合のみ。クラスタ内のすべての Integration Server

パラメータ	指定内容
	でタスクを実行する必要がある場合は、[**すべてのサーバ**] を選択します。ステートレスクラスタのサーバについては、このオプションは使用できません。 ■ 特定のサーバでのみタスクを実行する場合は、データベースに接続する Integration Server のリストから名前を選択します。 デフォルトは現行 Integration Server です。 [**ターゲットノード**] オプションの詳細については、779 ページの「[ターゲットノード] オプション」を参照してください。

5. [**タスクの実行が遅延している場合**] で実行するアクションを選択します。

パラメータ	Integration Server の処理
[即時実行]	タスクがどれだけ遅れているかに関係なく、即時にタスクを実行します。
[スキップして次にスケジュールされているタイミングで実行]	今回はタスクの実行を省略し、次回のスケジュール時間に再度タスクを実行します。1 回だけ実行されるタスクについては、このオプションは使用できません。
[一時停止]	管理者がタスクを再開またはキャンセルするまで、タスクを一時停止状態にします。

Integration Server はスケジュール済みタスクの状況を定期的にチェックします。スケジュール時間を過ぎているのに開始されていないタスクが検出された場合、遅れているタスクに関する特別なアクションが指定されていない限り、Integration Server はタスクを即時に実行します。ユーザが [(***xxx* 分以上遅延している場合の処理**)] フィールドで指定した時間だけスケジュールされた開始時刻から遅れているタスクについては、Integration Server でこの「遅延アクション」が実行されます。

注:
[(***xxx*分以上遅延している場合の処理**)] フィールドに指定できる最大分数は、35000 です。

遅れているが、指定された期間を過ぎていないタスクは、Integration Server で即時に実行されます。

6. Integration Server でサービスを実行する時間と頻度を指定するには、[**1 回だけ実行**]、[**繰り返し**]、[**複雑な繰り返し**] のいずれかのオプションを選択します。

選択項目	目的
[1 回だけ実行]	1 回だけ実行するようにタスクをスケジュールします。以下を指定します。 ■ [**日時**]Integration Server でサービスを実行する日付。 YYYY/MM/DD の形式で日付を入力します。たとえば、2010 年 3 月 11 日にサーバでサービスが実行されるように設定する場合、「`2010/03/11`」と指定します。 ■ [**時間**]Integration Server でサービスを実行する時刻。 HH:MM:SS 形式 (24 時間形式) を使用して時刻を入力します。たとえば、1:00:00 a.m. に Integration Server でサービスが実行されるように設定する場合、「`1:00:00`」と指定し、1:00:00 p.m. に Integration Server で実行されるように設定する場合は、「`13:00:00`」と指定します。
[繰り返し]	単純な繰り返しタスク (指定した時刻に 1 日 1 回など) をスケジュールします。以下を指定します。 ■ [**開始日**] および [**開始時刻**]最初の実行の日時。 [**開始日**] の指定には、YYYY/MM/DD 形式を使用します。[**開始時刻**] の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。 たとえば、2010 年 5 月 3 日 1:00:00 p.m にサービスの最初の実行が開始されるように設定する場合、[**開始日**] に「`2010/05/03`」と指定し、[**開始時刻**] に「`13:00:00`」と指定します。 ■ [**終了日**] および [**終了時刻**]最後の実行の日時。 [**終了日**] の指定には、YYYY/MM/DD 形式を使用します。[**終了時刻**] の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。 たとえば、2010 年 6 月 4 日の 02:00:00 a.m にサービスの実行を終了するように設定する場合、[**終了日**] に「2010/06/04」と指定し、[**終了時刻**] に「02:00:00」と指定します。 [**終了日**] を指定しなかった場合、Integration Server では現在の日付が適用されます。[**終了時刻**] を指定しなかった場合、Integration Server では現在時刻が適用されます。[**終了日**] および [**終了時刻**] を指定しなかった場合、Integration Server でサービスは無期限に実行されるか、スケジュールされているユーザタスクがキャンセルされるまで実行されます。

選択項目	目的
	■ [**間隔**]実行間隔。 Integration Server によるサービス実行の間隔 (秒単位) を入力します。 ■ [**完了後に繰り返す**]前回のサービスの終了を待ってから次の実行を開始するかどうか。 たとえば、GetData というサービスは毎分実行されるようにスケジュールされているが、1 回の GetData サービスの完了までに 1 分以上かかる場合があるとします。デフォルトでは、前回の実行がまだ完了していない場合でも、次の実行が開始されるように Integration Server が設定されています。 [**完了後に繰り返す**] チェックボックスをオンにしておくと、Integration Server では前回のサービスの実行が完了するのを待機してから、次回の実行が開始されます。前回のサービスが実行中のために開始できなかった場合、次回の実行は遅延します。 [繰り返し] オプションの使用の詳細については、775 ページの「単純な [繰り返し] オプション」を参照してください。
[複雑な繰り返し]	より複雑な間隔で繰り返す (1 日おき、月に 2 回など) タスクをスケジュールします。以下を指定します。 ■ [**開始日**] および [**開始時刻**]最初の実行の日時。 [**開始日**] の指定には、YYYY/MM/DD 形式を使用します。[**開始時刻**] の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。 たとえば、2010 年 5 月 3 日 1:00:00 p.m にサービスの最初の実行が開始されるように設定する場合、[**開始日**] に「`2010/05/03`」と指定し、[**開始時刻**] に「`13:00:00`」と指定します。[**開始日**] を指定しなかった場合、[**実行マスク**] パラメータに指定されている開始日に最初の実行が開始されます。[**開始時刻**] を指定しなかった場合は、現在時刻が適用されます。 ■ [**終了日**] および [**終了時刻**]最後の実行の日時。 [**終了日**] の指定には、YYYY/MM/DD 形式を使用します。[**終了時刻**] の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。 たとえば、2010 年 6 月 4 日の 02:00:00 a.m にサービスの実行を終了するように設定する場合、[**終了日**] に「2010/06/04」と指定し、[**終了時刻**] に「02:00:00」と指定します。

選択項目	目的
	[**終了日**] を指定しなかった場合、Integration Server では現在の日付が適用されます。[**終了時刻**] を指定しなかった場合、Integration Server では現在時刻が適用されます。[**終了日**] および [**終了時刻**] を指定しなかった場合、Integration Server でサービスは無期限に実行されるか、スケジュールされているユーザタスクがキャンセルされるまで実行されます。 ■ [**実行マスク**]タスクを実行する月、日 (1～31)、曜日 (日曜日～月曜日)、時、分、秒を指定します。各カテゴリから 1 つ以上の項目を選択できます。複数の項目を選択するには、Ctrl キーを押しながら選択します。カテゴリから項目を選択しない場合、Integration Server ではすべての項目が選択されていると想定されます。 ■ [**完了後に繰り返す**]前回のサービスの終了を待ってから次の実行を開始するかどうか。 たとえば、GetData というサービスは毎分実行されるようにスケジュールされているが、1 回の GetData サービスの完了までに 1 分以上かかる場合があるとします。デフォルトでは、前回の実行がまだ完了していない場合でも、次の実行が開始されるように Integration Server が設定されています。 [**完了後に繰り返す**] チェックボックスをオンにしておくと、前回のサービスの実行が完了するのを待機してから、次回の実行が開始されます。前回のサービスが実行中のために開始できなかった場合、次回の実行は遅延します。 [複雑な繰り返し] オプションの使用の詳細については、776 ページの「[複雑な繰り返し] オプション」を参照してください。

注:
DST (Daylight Saving Time : 夏時間) に対する移行があるため、Integration Server はスケジュール済みタスクをスキップまたは繰り返しません。Integration Server が DST に対する移行を処理する方法の詳細については、781 ページの「夏時間に対する移行がスケジュール済みタスクに及ぼす影響」を参照してください。

7. [**タスクの保存**] をクリックします。

スケジュールされているユーザタスクの表示

注:
同じデータベースに接続したサーバのグループの一部として Integration Server が実行されている場合は、そのデータベースに接続したすべての Integration Server の [**サーバ**] > [スケジューリング] ページで、すべてのタスクを確認できます。

スケジュールされているユーザタスクを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [スケジューリング] に移動します。

   [スケジューラ] ページに、スケジュールされているユーザタスクのリストが表示されます。Integration Server によって、[スケジューラ] ページから期限切れのスケジュール済みユーザタスクが自動的に削除されます。次のサーバ再起動まで Integration Server で有効期限切れのタスクが削除されないようにするには、watt.server.scheduler.deleteCompletedTasks パラメータを「false」に設定します。このパラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

スケジュール済みタスクのリストのフィルタ処理

デフォルトでは、[**サーバ**] > [スケジューリング] ページに、実行するスケジュールが設定されたユーザ定義のタスクがすべて表示されます。フィルタを使用し、表示されるタスクを制限することで、管理しやすい簡潔なリストを作成できます。

スケジュール済みタスクのリストをフィルタ処理するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [スケジューリング] に移動します。

   ユーザ定義のタスクのリストが表示されます。

3. [**サービスのフィルタ**] をクリックします。ユーザタスクのリストの上部にフィルタ処理オプションが表示されます。

   注:
   [**サービスのフィルタ**] が有効化されている間は、Integration Server に対するすべての変更 (新規タスクなど) はスケジュール済みタスクのリストに反映されません。[**すべてのサービスを表示**] をクリックして通常モードに戻ると、リストが更新されます。

4. 以下のオプションの一部またはすべてを選択します。

オプション	説明
[サービス名]	フィルタに対して指定するストリング。フィルタ条件には、リテラルや、リテラルとワイルドカード文字の組み合わせを使用できます。ワイルドカード文字としてサポートされているのは、「*」 (アスタリスク) と「?」 (疑問符) だけです。フィルタ条件を空白にした場合は、すべてのサービスが結果に含められます。

オプション	説明
	[**サービス名**] フィールドに入力されたサービス名の大文字と小文字は区別されます。たとえば、「Wm*」と入力した場合、「wm」で開始するサービスはすべて無視されます。
[**状態がアクティブ**]	アクティブなサービスのみを表示するかどうかを制御します。オフになっている場合、すべての状態のサービスが表示されます。
[**リモートタスクの非表示**]	現在のサーバ上で実行されるタスクのみを表示するかどうかを制御します。[**ターゲットノード**] が、クラスタ内の [**すべてのサーバ**] またはデータベースに接続した [**任意のサーバ**] で実行されるように設定されている場合に、現在のサーバで実行されるようにスケジュールされたタスクも、この対象となります。他のサーバ上でのみ実行されるようにスケジュールされたタスクは表示されません。

スケジュールされているユーザタスクの更新

注:
Integration Server がクラスタの一部である場合に、クラスタ内のすべての Integration Server 上で実行するようにスケジュールされているタスクの特性を更新するには、親タスクを変更する必要があります。クラスタ内のすべてのサーバで実行されるスケジュール済みタスクの詳細については、779 ページの「[ターゲットノード] オプション」を参照してください。

スケジュールされているユーザタスクを更新するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [スケジューリング] に移動します。
3. 更新するユーザタスクのサービス名をクリックします。
4. 選択したユーザタスクのスケジュールオプションを更新します。指定可能なオプションの詳細については、764 ページの「ユーザタスクのスケジュール」を参照してください。
5. [**タスクの更新**] をクリックします。

ユーザタスクの一時停止

Integration Server では単一のユーザタスクまたはすべてのスケジュール済みユーザタスクを一時停止できます。

単一ユーザタスクの一時停止

ユーザタスクを一時停止した場合、スケジュール自体は残りますが、実行の再開を指定するまでは実行されません。一時停止中に期限切れとなったタスクには、期限切れのマークが付けられます。

注:
Integration Server がクラスタの一部である場合、クラスタ内のすべての Integration Server 上で実行するようにスケジュールされているタスクを一時停止するときには、個々の子タスクを一時停止するか、親タスクを一時停止することですべてのタスクを一度に一時停止することができます。クラスタ内のすべてのサーバで実行されるスケジュール済みタスクの詳細については、779 ページの「[ターゲットノード] オプション」を参照してください。

スケジュールされているユーザタスクを一時停止するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [スケジューリング] に移動します。
3. [サービス] 一覧でタスクを見つけ、[状態] 列で ✓ アイコンをクリックしてタスクを一時停止します。タスクを一時停止するかどうかを確認するページが表示されます。[**OK**] をクリックします。✓ アイコンが [一時停止中] に置き換えられ、タスクが一時停止されたことが示されます。

すべてのユーザタスクの一時停止

タスクを一時停止すると、ユーザタスクのみに影響し、システムタスクには影響ありません。

Integration Server ですべてのスケジュール済みユーザタスクを一時停止すると、次のようになります。

- Integration Server は、どのスケジュール済みユーザタスクも開始しません。ただし、新規タスクをスケジュールして、他のタスク固有アクションを実行することはできます。
- 現在実行されているスケジュール済みユーザタスクは実行が継続されます。
- 一時停止アクションは、システムタスクには影響せず、実行が継続されます。

スケジュールされているすべてのユーザタスクを一時停止するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [スケジューリング] に移動します。
3. [**スケジューラの一時停止**] をクリックします。タスクを一時停止するかどうかを確認するメッセージが Integration Server で表示されます。[**OK**] をクリックします。

Integration Server では、スケジュール済みのユーザタスクを再開した後でのみ、そのタスクを開始します。

一時停止中のユーザタスクの再開

一時停止したユーザタスクのすべての実行を再開できます。Integration Server ではすべてのスケジュール済みユーザタスクの実行を再開することもできます。

一時停止中のユーザタスクの再開

一時停止されているタスクのすべてのスケジュールされている実行を再開するには、以下の手順に従います。

注:
Integration Server がクラスタの一部である場合、クラスタ内のすべての Integration Server 上で実行するようにスケジュールされているタスクを再開するときには、個々の子タスクを再開するか、親タスクを再開することですべてのタスクを一度に再開することができます。クラスタ内のすべてのサーバで実行されるスケジュール済みタスクの詳細については、779ページの「[ターゲットノード] オプション」を参照してください。

一時停止中のユーザタスクの実行を再開するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [スケジューリング] に移動します。
3. [**サービス**] の一覧からタスクを選択し、[アクティブ] 列の [**一時停止中**] をクリックして、タスクをアクティブにします。タスクを再開するかどうかを確認するページが表示されます。[**OK**] をクリックします。

   [一時停止中] が [アクティブ] アイコン ✓ に置き換えられ、タスクが再び実行可能になったことが示されます。

すべての一時停止中のユーザタスクの再開

Integration Server ですべてのスケジュール済みユーザタスクの実行を再開すると、次のようになります。

- Integration Server は、スケジュール済みユーザタスクの開始を再開します。
- タスクが一時停止したときに実行中だったスケジュール済みユーザタスクがまだ実行中である場合は、その実行が継続されます。
- [状態] 列の ✓ アイコンをクリックして一時停止させた各タスクは、明示的にアクティブにしない限り、一時停止したままになります。

スケジュールされているすべてのユーザタスクを再開するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [スケジューリング] に移動します。

3. [**スケジューラの再開**] をクリックします。タスクの開始を再開するかどうかを確認するメッセージが Integration Server で表示されます。[**OK**] をクリックします。

   Integration Server は、スケジュール済みユーザタスクを開始します。

   重要:
   スケジュール済みの実行時刻の時点でタスクの実行が一時停止されていたために、タスクの実行時刻が過ぎてしまった場合は、次回のタスク実行は [**タスクの実行が遅延している場合**] 設定によって異なります。この設定に基づいて、Integration Server は、タスクの開始を再開するとすぐにこのタスクを開始するか、今回のタスクの実行をスキップするか、またはタスクを一時停止して管理者のアクションを待ちます。詳細については、764 ページの「ユーザタスクのスケジュール」を参照してください。

スケジュールされているユーザタスクのキャンセル

スケジュールされているタスクをキャンセルすると、そのタスクはジョブキューを含むデータベースから永続的に消去されます。

注:
サーバがクラスタの一部である場合、クラスタ内のすべてのサーバ上で実行するようにスケジュールされているタスクをキャンセルするときには、個々の子タスクをキャンセルするか、親タスクをキャンセルすることですべてのタスクを一度にキャンセルすることができます。クラスタ内のすべてのサーバで実行されるスケジュール済みタスクの詳細については、779 ページの「[ターゲットノード] オプション」を参照してください。

スケジュールされているユーザタスクをキャンセルするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [スケジューリング] に移動します。

3. キャンセルするユーザタスクの [削除] 列にある ✕ アイコンをクリックします。ユーザタスクをキャンセルするかどうかを確認するダイアログが表示されます。[**OK**] をクリックします。

スケジュールされているシステムタスクの表示

Integration Server はシステムタスク (期限切れになるセッションのなど) を定期的に実行する必要があります。Integration Server では、これらのタスクが自動的にスケジュールされます。

スケジュールされているシステムタスクを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [スケジューリング] に移動します。

3. [**システムタスクの表示**] をクリックします。

   [**システムタスクの表示**] ページが表示されます。この画面には、スケジュールされている各タスクの名前、タスクが次回実行される日付と時刻、およびタスクが実行される頻度 ([**間隔**]) が一覧表示されます。

   注:
   [**システムタスクの表示**] ページには、ローカルサーバのタスクのみが表示されます。同じデータベースに接続したサーバのグループを実行している場合、そのデータベースに接続した他のサーバのシステムタスクは表示されません。

単純な [繰り返し] オプション

単純な [繰り返し] オプションを使用すると、ユーザが指定した間隔でサービスが繰り返されます。

設定項目	説明
[開始日]	サーバでサービスを最初に実行する日付。日付の指定には、YYYY/MM/DD 形式を使用します。このフィールドを空白にすると、現在の日付にタスクが開始されます。
[開始時刻]	サーバでサービスの実行が開始される時刻。時刻の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。このフィールドを空白にすると、タスクは即時に開始されます。
[終了日]	サーバでサービスを最後に実行する日付。日付の指定には、YYYY/MM/DD 形式を使用します。 [**終了日**] を空白にすると、サーバは現在の日付を [**終了日**] として使用します。 [**終了日**] および [**終了時刻**] フィールドを空白にすると、Integration Server でサービスは無期限に実行されるか、スケジュールされているユーザタスクがキャンセルされるまで実行されます。
[終了時刻]	終了日にサーバでサービスを実行する時刻。時刻の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。 [**終了時刻**] フィールドを空白にすると、サーバは現在の時刻を [**終了時刻**] として使用します。 [**終了日**] および [**終了時刻**] フィールドを空白にすると、サービスは無期限に実行されるか、スケジュールされているユーザタスクがキャンセルされるまで実行されます。

設定項目	説明
[繰り返し]/[完了後に繰り返す]	Integration Server が前回のサービスの終了を待ってから次の実行を開始するかどうか。 たとえば、GetData というサービスは毎分実行されるようにスケジュールされているが、1 回の GetData サービスの完了までに 1 分以上かかる場合があるとします。デフォルトでは、前回の実行がまだ完了していない場合でも、次の実行が開始されるように設定されています。[**完了後に繰り返す**] チェックボックスをオンにしておくと、Integration Server では前回のサービスの実行が完了するのを待機してから、次回の実行が開始されます。前回のサービスが実行中のために開始できなかった場合、次回の実行は遅延します。
[間隔]	サービスの実行間隔 (秒単位)。たとえば、24 時間ごとにサービスが実行されるようにする場合は、実行間隔を「`86400`」秒に指定します。

次の表に、[単純な繰り返し] オプション設定の使用方法の例を示します。

サービスの実行	設定項目	指定内容
2010 年の 7 月 1 日の毎時に実行。	[開始日]	2010/07/01
	[開始時刻]	0:00:00 AM
	[終了日]	2010/07/01
	[終了時刻]	0:00:00 AM
	[間隔]	3600

[複雑な繰り返し] オプション

[**複雑な繰り返し**] オプションを指定すると、サービスはユーザが指定した複雑な間隔に基づいて繰り返し実行されます。このオプションを使用して、サービスが実行されるタイミングを柔軟に指定できます。

実行されるサービスの時間と頻度を指定するには、次の表に説明する設定を組み合わせて使用します。

設定項目	説明
[開始日]	サーバでサービスを最初に実行する日付。日付の指定には、YYYY/MM/DD 形式を使用します。このフィールドを空白にすると、タスクは他の設定項目に従って、指定されている最初の日付に実行されます。
[開始時刻]	サーバでサービスの実行が開始される時刻。時刻の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。このフィールドが空白の場合は、現在の時刻が使用されます。

設定項目	説明
[終了日]	サーバでサービスを最後に実行する日付。日付の指定には、YYYY/MM/DD 形式を使用します。 [**終了日**] を空白にすると、サーバは現在の日付を [**終了日**] として使用します。 [**終了日**] および [**終了時刻**] フィールドを空白にすると、Integration Server でサービスは無期限に実行されるか、スケジュールされているユーザタスクがキャンセルされるまで実行されます。
[終了時刻]	終了日にサーバでサービスを実行する時刻。時刻の指定には、HH:MM:SS 形式 (24 時間形式) を使用します。 [**終了時刻**] フィールドを空白にすると、サーバは現在の時刻を [**終了時刻**] として使用します。 [**終了日**] および [**終了時刻**] フィールドを空白にすると、サービスは無期限に実行されるか、スケジュールされているユーザタスクがキャンセルされるまで実行されます。
[繰り返し]/[完了後に繰り返す]	Integration Server が前回のサービスの終了を待ってから次の実行を開始するかどうか。 **注:** Integration Server は、途中まで終了したサービス実行を完了したタスクとして数えません。完了する前にタスクが失敗した場合、Integration Server はタスクを遅延とみなします。タスクの次回実行は、アクションの [**タスクの実行が遅延している場合**] オプションの設定によって異なります。詳細については、以下を参照してください。 764 ページの「ユーザタスクのスケジュール」 Integration Server で前回のサービスの実行が完了するのを待機してから次回の実行が開始されるように設定する場合、このチェックボックスをオンにします。 たとえば、GetData というサービスは毎分実行されるようにスケジュールされているが、1 回の GetData サービスの完了までに 1 分以上かかる場合があるとします。デフォルトでは、前回の実行がまだ完了していない場合でも、次の実行が開始されるように Integration Server が設定されています。[**完了後に繰り返す**] チェックボックスをオンにしておくと、Integration Server では前回のサービスの実行が完了するのを待機してから、次回の実行が開始されます。前回のサービスが実行中のために開始できなかった場合、次回の実行は遅延します。
[実行マスク]	サービスを実行する月、日 (1～31)、曜日 (日曜日～土曜日)、時、分を指定します。 各カテゴリから 1 つ以上の項目を選択できます。複数の項目を選択するには、Ctrl キーを押しながら選択します。カテゴリから項目を選択し

設定項目	説明
	ない場合、Integration Server ではすべての項目が選択されていると想定されます。たとえば、[月] を指定しなかった場合でも、Integration Server ではサービスを毎月実行するものと見なされます。設定項目のいずれかを選択しなかった場合でも、Integration Server では毎月、毎日、すべての曜日、毎時、毎分にサービスを実行するものと見なされます。つまり、タスクを追加した時点からサービスが毎分実行されます。

Integration Server でサービスが実行される日付と時刻は、選択したすべての設定を組み合わせて決定されます。

次の表に、[**複雑な繰り返し**] オプション設定の使用方法の例を示します。

サービスの実行	設定項目	指定内容
2010 年の毎月 28 日の午前零時に実行。	[開始日]	2010/01/01
	[開始時刻]	0:00:00 AM
	[終了日]	2010/12/31
	[終了時刻]	0:00:00 AM
	[月]	選択しない
	[日]	28
	[曜日]	選択しない
	[時]	0
	[分]	0
1 月、2 月、および 3 月の毎週月曜日の 2:30 p.m. に実行。期間は不確定。	[開始日]	空白
	[開始時刻]	空白
	[終了日]	空白
	[終了時刻]	空白
	[月]	1 月、2 月、3 月
	[日]	選択しない
	[曜日]	月曜日
	[時]	14
	[分]	30

サービスの実行	設定項目	指定内容
2010 年 6 月の毎週火曜日の毎時に実行。	[開始日]	2010/06/01
	[開始時刻]	00:00:00 (または空白)
	[終了日]	2010/06/30
	[終了時刻]	00:00:00 (または空白)
	[月]	6 月
	[日]	選択しない
	[曜日]	火曜日
	[時]	選択しない
	[分]	0
2010 年 6 月の毎週火曜日の毎時毎分に実行。	[開始日]	2010/06/01
	[開始時刻]	00:00:00 (または空白)
	[終了日]	2010/06/30
	[終了時刻]	00:00:00 (または空白)
	[月]	6 月
	[日]	選択しない
	[曜日]	火曜日
	[時]	選択しない
	[分]	選択しない

[ターゲットノード] オプション

同じデータベースに接続したサーバのグループを実行している場合に、どのサーバでタスクを実行するかを制御できます。ステートフルクラスタでは、クラスタ内の 1 つ、任意、またはすべての Integration Server で実行するジョブをスケジュールできます。クラスタで実行するジョブについては、サーバはクラスタリングに対して有効になっている必要があります。また、既存のジョブはクラスタで実行するようフラグ設定されている必要があります。ステートレスクラスタでは、クラスタ内の 1 つまたは任意の (ただし、すべてではない) Integration Server で実行するジョブをスケジュールできます。

次の表は、スケジュールされたタスクに対するターゲットノードオプションを示しています。

設定項目	説明
<specific_server>	タスクはユーザが指定したサーバ上で実行されます。
[任意のサーバ]	タスクは、データベースに接続した任意のサーバで実行されます。タスクを 1 つのサーバだけで実行する場合、どのサーバで実行してもかまわないときには、このオプションを使用します。たとえば、クラスタ環境 (ステートフルクラスタ) では、クラスタ内のすべてのサーバが共有するパーツ在庫アプリケーションの 1 つのデータベースに対して特定の機能を毎日 1 回実行する必要がある場合は、クラスタ内のどのサーバでもこの機能を実行できます。クラスタリングが有効化されているときには、[**任意のサーバ**] オプションがデフォルト設定です。 **注:** [**任意のサーバ**] オプションでは、タスクの実行に使用されるサーバの順序は指定されません。つまり、負荷分散は実行されません。その代わり、データベースに接続した個々のサーバ上でスケジューラインスタンスが 1 つずつ実行されます。各スケジューラインスタンスは、スケジュールされているジョブに関する情報が保存されているデータベースを定期的にチェックします。実行開始時間となったタスクを検出した最初のスケジューラインスタンスがタスクを実行し、スケジュール済みジョブに関する情報が保存されているデータベース内のタスクに完了のマークを付けます。これにより、データベースに接続した他のサーバ上で実行されているスケジューラインスタンスは、このタスクを実行する必要がないことを理解します。サードパーティ製の負荷分散機能をインストールした場合でも、この動作は変わりません。
[すべてのサーバ]	クラスタ化された Integration Server の場合のみ。[**すべてのサーバ**] を選択すると、クラスタ内のすべてのサーバでタスクが実行されます。たとえば、クラスタ内のすべてのサーバ上で実行される 1 つのアプリケーションについて、各サーバがアプリケーション用の独自のデータベースを保持しているとします。すべてのデータベースに対して毎日クリーンアップタスクを実行する必要がある場合、1 つのサーバから、クラスタ内のすべてのサーバ上で毎日実行するようにタスクをスケジュールできます。 クラスタ内のすべてのサーバ上でタスクを実行するようにスケジュールした場合、クラスタ内の各サーバのタスクは、メイン (親) タスクと子タスクに分割されます。これらのタスクの詳細については、780 ページの「クラスタ環境内のタスク」を参照してください。 ステートレスクラスタの Integration Server では、このオプションは使用できません。

クラスタ環境内のタスク

クラスタ内のすべてのサーバ上でタスクを実行するようにスケジュールした場合、クラスタ内の各サーバのタスクは、メイン (親) タスクと子タスクに分割されます。

親タスクの状態と子タスクの状態が異なる場合があります。たとえば、親タスクの状態がアクティブ、1つの子タスクがアクティブ、別の子タスクが一時停止という状況も考えられます。一般的に、親タスクの状態は、少なくとも 1 つの子タスクがアクティブまたは実行中の場合はアクティブになり、すべての子タスクが一時停止の場合は一時停止、すべての子タスクが期限切れの場合は期限切れになります。

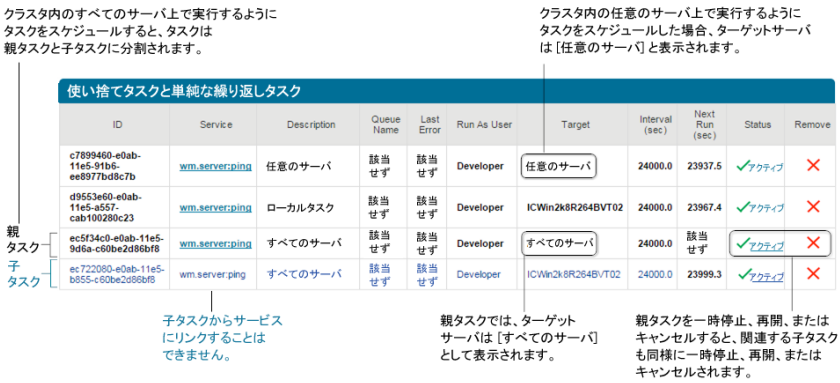
個々の子タスクについては、一部のアクション (アクティブ化、一時停止) を個別に実行できますが、タスクの特性を変更したい場合は、親タスクを通じて変更する必要があります。

親タスクと子タスクの同期状態を維持するために、次の場合は Integration Server によって子タスクが自動的に更新されます。

- 親タスクを変更した場合。たとえば親タスクの終了日を変更すると、Integration Server は関連する子タスクで終了日を自動的に変更します。
- クラスタノードを追加または削除した場合。たとえば、別の Integration Server をクラスタに追加すると、Integration Server は Integration Server の開始時に、そのノードの子タスクを自動的に作成します。
- 子タスクを削除した場合。たとえば、まだ存在するクラスタノードの子タスクを削除すると、Integration Server は Integration Server の開始時、または親タスクの更新時に、そのノードの子タスクを自動的に再作成します。

次の図は、Integration Server Administrator の [**サーバ**] > [スケジューリング] ページで親タスクと子タスクがどのように表示されるかを示しています。

クラスタ環境内のタスク



注:
Integration Server のクラスタで [スケジューラ] を使用する場合、すべてのサーバのシステムクロックを同期することで、複雑な繰り返しタスクを指定の時間に必ず実行させることができます。

夏時間に対する移行がスケジュール済みタスクに及ぼす影響

夏時間 (DST) に対する移行は、次のように Integration Server のスケジュール済みタスクに影響を及ぼします。

- DST に移行すると、システムクロック時間が 1 時間早く進みます。ただし、DST に移行したためにスキップした時間中に実行するようにスケジュールされたスケジュール済みタスクについて、Integration Server はスキップしません。タスクがスキップされた時間中に実行するようにスケジュールされていた場合、Integration Server はそのタスクをその後の時間に実行します。たとえば、タスクを午前 1:30:00 に実行するようにスケジュールした場合、システムクロック時間が午前 1:00:00 に DST に移行すると、Integration Server はそのタスクをシステムクロック時間の午前 2:30:00 に処理します。
- DST から移行すると、システムクロック時間が 1 時間戻されます。DST との重複時間に実行がスケジュールされた繰り返しタスクの場合、Integration Server は以下のいずれかを実行します。
  - タスクに時間マスクが含まれている場合 (たとえば、毎日 0115 に実行する場合は「1:15」)、タスクは重複前の (最初の) 一致時刻に 1 回だけ実行されます。たとえば、タスクが 1:15:00 に実行される場合、システムクロック時間が午前 2:00:00 に DST に移行して午前 1:00:00 に戻された場合、Integration Server はそのタスクを 1:15:00 に繰り返しません。
  - タスクに時間マスクがない場合 (たとえば、毎時 15 分過ぎに実行を意味する「hh:15」)、タスクは重複前の時刻 (最初の 1:15) に実行され、重複時刻 (2 番目の 1:15) にも実行されます。
- 単純および複雑な繰り返しタスクは、DST に対する移行による影響を受けません。Integration Server は、繰り返しタスクを指定された間隔で処理します。つまり、タスクが 1 時間ごとに実行するようにスケジュールされている場合、Integration Server はシステムクロック時間に関係なく、1 時間間隔でタスクを実行します。

40 サービス結果のキャッシング

- キャッシングとは 784
- キャッシュされた結果が返るタイミング 784
- サービス結果のキャッシングにパブリックキャッシュを使用する 785
- キャッシュのリセット 786
- サービスキャッシュの使用状況の監視 787
- パブリックキャッシュのサービス結果の表示 788

キャッシングとは

キャッシングは、サービスのパフォーマンスを向上させるための最適化機能です。

Software AG Designer のキャッシングを使用するサービスを指定します。サービスのキャッシュを有効にすると、Integration Server は指定した期間はローカルキャッシュまたは分散キャッシュ内のサービスを呼び出し、その後でパイプラインの内容全体を保存します。パイプラインには、キャッシュされたサービスで明示的に定義されている出力フィールド、およびフローの前のサービスによって生成された出力フィールドが含まれます。Integration Server が入力値セットが同一のサービス要求を続けて受信した場合、Integration Server はサービスを再度呼び出す代わりにキャッシュされた結果をクライアントに返します。

キャッシングによって、サービスの応答時間が著しく向上します。たとえば、大量トラフィックの商用 Web サーバなどの混雑しているデータソースから情報を抽出するサービスの場合、キャッシングは非常に便利です。Integration Server では、フロー、Java サービス、C/C++ サービスなど、あらゆる種類のサービスの結果をキャッシュできます。

キャッシングの目的は、データの流れとメモリ使用量とをうまく釣り合わせることです。キャッシュ効率を測定するには、Integration Server Administrator のサービスの統計を表示してパフォーマンスを監視します。また、キャッシング値は測定結果に基づいて調整できます。

Designer のサービスをキャッシングするための制御を設定します。手順については、『*webMethods Service Development Help*』を参照してください。

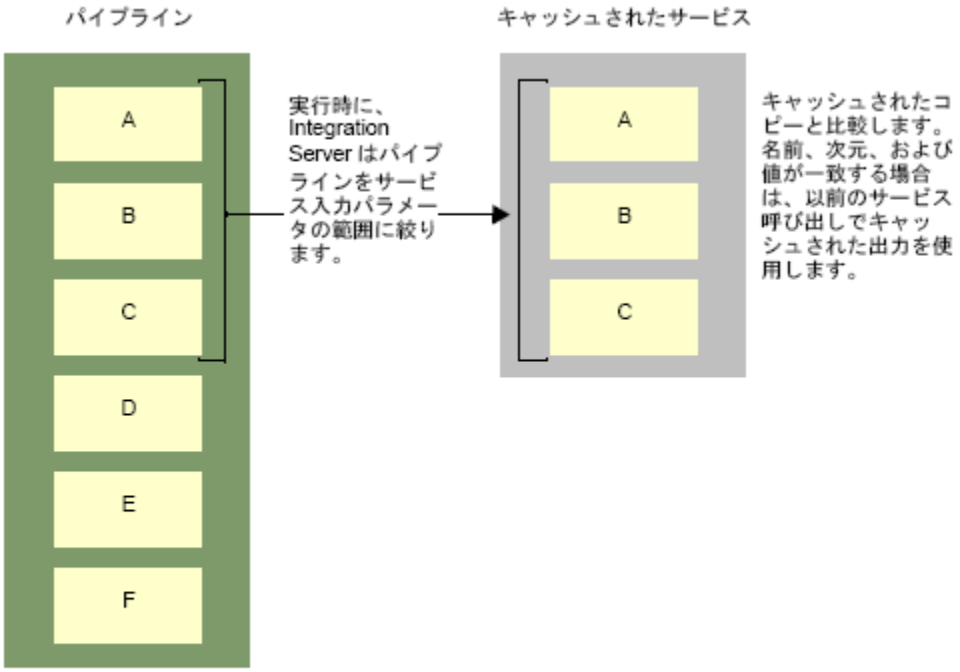
キャッシュされた結果が返るタイミング

Software AG Designer 内のサービスにキャッシングを有効にすると、サービス入力パラメータの有無によって Integration Server によるキャッシュ結果の処理方法が異なります。キャッシュされたサービスには入力パラメータを指定することを推奨します。

キャッシュされたサービスに入力パラメータがある場合、実行時に Integration Server がパイプラインを宣言されたサービス入力パラメータの範囲に絞ります。Integration Server は、範囲が限定された入力を以前に保存された入力のコピーと比較します。入力パラメータが同じ値のキャッシュエントリが存在する場合は、Integration Server は前のサービス呼び出しからキャッシュされた結果を返します。

注:
入力パラメータ値が現在の呼び出しと同一のキャッシュエントリがキャッシュに存在しない場合は、Integration Server は、サービスを実行してキャッシュ内に結果を保存します。

実行時に、キャッシュされたコピーと比較されるパイプライン入力



date/time サービスなどのようにキャッシュされたサービスに入力パラメータがなく、以前の結果がキャッシュに存在しない場合は、実行時に Integration Server がサービスを実行して結果を保存します。サービスを再度実行するときには、Integration Server はキャッシュされたコピーを使用します。つまり、Integration Server は現在のサービスの呼び出しにランタイムパイプラインを使用せず、キャッシュが期限切れになるまで、常にキャッシュ結果が返されます。

重要:
キャッシュされたサービス入力署名にドキュメント参照またはドキュメント参照リスト変数が含まれ、参照されているドキュメントタイプが変更または修正された場合は、サービスキャッシュをリセットする必要があります。リセットしないと、Integration Server はキャッシュ結果が期限切れになるまで、実行時にキャッシュされている古い入力パラメータを使用します。Designer または Integration Server Administrator からキャッシュをリセットできます。Integration Server Administrator からサービスキャッシュをリセットする詳細については、786 ページの「キャッシュのリセット」を参照してください。

サービス結果のキャッシングにパブリックキャッシュを使用する

デフォルトで、Integration Server はキャッシュされたサービス結果をローカルの ServiceResults システムキャッシュに保存します。このキャッシュは SoftwareAG.IS.Services システムキャッシュマネージャの一部です。複数の Integration Server がキャッシュされたサービス結果にアクセスできるようにする場合、代わりに分散キャッシュを含めたパブリックキャッシュ内にキャッシュされたサービス結果を保存できます。同じパッケージがインストールされている Integration Server のみ、分散キャッシュを共有できます。

サービス結果のキャッシングにパブリックキャッシュを使用する場合、以下の情報に留意してください。

- エレメントが一度キャッシュに置かれたら、期限切れにならないように、キャッシュを設定する必要があります。それには、キャッシュの [**エターナル**] チェックボックスをオンにします (835 ページの「キャッシュの作成」を参照)。
- キャッシュされたサービス結果の経過時間は、サービスのキャッシュ期限切れプロパティ値および watt.server.cache.flushMins サーバ設定パラメータ値によって異なります。
- デフォルトで、Integration Server はサービスのキャッシュされた結果を返すとき、キャッシュされた結果の実際値を返します。Integration Server が実際値の代わりに参照を返すようにするには、キャッシュの [**読み取り時にコピー**] および [**書き込み時にコピー**] チェックボックスをオフにします (835 ページの「キャッシュの作成」を参照)。
- サービス結果のキャッシングに使用されるパブリックキャッシュが無効になっているか、サービス結果のキャッシングに使用されるキャッシュを含むパブリックキャッシュマネージャがシャットダウンされている場合、Integration Server はサービス結果のキャッシュまたは取得を行いません。代わりに、Integration Server はサービスを実行します。
- サービス結果のキャッシングに使用されるパブリックキャッシュが有効にされたか、サービス結果のキャッシングに使用されるキャッシュを含むパブリックキャッシュマネージャが開始された場合、Integration Server はキャッシュを再初期化します。

パブリックキャッシュの作成、パブリックキャッシュマネージャ、および分散キャッシュの詳細については、799 ページの「Integration Server での Terracotta Ehcache の設定 」を参照してください。

注:
Integration Server Administrator の [**サーバ**] > [サービスの使用状況] ページにはサービス結果に関する統計情報が表示されます。分散キャッシュの場合、現在の Integration Server インスタンスのみの統計情報になります。Integration Server Administrator は、サービス結果のキャッシングに同じ分散キャッシュを使用して、すべての Integration Server の集約された統計情報を提供しません。

Integration Server がサービス結果のキャッシングにパブリックキャッシュを使用するよう設定するには、サーバ設定パラメータ `watt.server.serviceResults.cache` および `watt.server.serviceResults.cacheManger` のそれぞれにキャッシュとキャッシュマネージャの名前を指定します。

キャッシュのリセット

すべてのサービスに対するキャッシュをリセットすることも、特定のサービスに対するキャッシュをリセットすることもできます。

注:
Integration Server は、サービスが編集されると常にサービスのキャッシュを自動的にリセットします。ただし、入力署名にドキュメント参照変数が含まれ、参照されるドキュメントタイプが変更された場合は、サービスキャッシュをリセットする必要があります。リセットしないと、Integration Server はキャッシュ結果が期限切れになるまで、実行時にキャッシュされている古い入力パラメータを使用します。

すべてのサービスに対するキャッシュのリセット

Integration Server がすべてのサービスに対するキャッシュをリセットすると、すべてのサービスに対してキャッシュされたサービス結果はすべてメモリから削除されます。

また、pub.cache.serviceResults:resetServerCache サービスを使用して、すべてのサービスのキャッシュをリセットすることもできます。

すべてのサービスに対するキャッシュをリセットするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [サービスの使用状況] に移動します。
3. [**サーバキャッシュのリセット**] をクリックして、表示されているすべてのサービスのキャッシュをリセットします。

特定のサービスに対するキャッシュのリセット

Integration Server がサービスに対するキャッシュをリセットすると、そのサービスに対してキャッシュされたサービス結果はメモリから削除されます。

また、pub.cache.serviceResults:resetServiceCache サービスを使用して、特定のサービスのキャッシュをリセットすることもできます。

特定のサービスに対するキャッシュをリセットするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [サービスの使用状況] に移動します。
3. キャッシュをリセットするサービスの名前を選択します。そのサービスの情報ページが表示されます。
4. [**サービスキャッシュのリセット**] をクリックします。

サービスキャッシュの使用状況の監視

サービスキャッシュのパフォーマンスを監視するには、以下の手順に従います。

キャッシュのパフォーマンスを監視するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [サービスの使用状況] に移動します。

[**サービスの使用状況**] ページに、キャッシュ制御された各サービスに関して、キャッシュ制御設定の現在の結果が表示されます。デフォルトで、Integration Server Administrator にはサービスがアルファベット順に表示されます。

ヒント:
[**実行中のサービスを一番上に表示する**] チェックボックスをオンにすると、Integration Server で現在実行中のすべてのサービスをページの上部にまとめて表示できます。現在実行中のサービスについては、サービス名の右側にカッコで囲まれた数字が表示されます。この数字は、サービスの実行中のインスタンスの数を示しています。

パブリックキャッシュのサービス結果の表示

特定のサービスのキャッシュされたサービス結果を表示するには、以下の手順に従います。パブリックキャッシュに保存されたキャッシュ済みサービス結果は表示できますが、ServiceResults システムキャッシュのサービス結果は表示できません。

また、pub.cache.serviceResults:listServiceCache サービスを使用して、特定のサービスのキャッシュされたサービス結果をリストすることもできます。この手順またはサービスを使用する前に、サービス結果キャッシュを検索可能にし、キーの自動インデックスの許可を設定します。それには、[**検索可能**] チェックボックス、および [**自動インデックスを許可**] の横にある [**キー**] チェックボックスをオンにします (835 ページの「キャッシュの作成」を参照)。

注:
サービス結果のキャッシュの内容を公開すると、セキュリティ上のリスクが伴うことがあります。

サービスのサービス結果のキャッシュを表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. [**パッケージ**] リストで、キャッシュされたサービス結果を表示する対象のサービスが入ったパッケージをクリックします。
4. [***packageName* 内のサービスを表示**] をクリックします。
5. [**サービス**] リストで、キャッシュされたサービス結果を表示する対象のサービスをクリックします。
6. [**パッケージ**] > [**管理**] > [***packageName***] > [**サービス**] > [***serviceName***] に移動し、[**サービスキャッシュの一覧**] をクリックします。

   Integration Server はサービスのキャッシュされた結果のリストを表示します。Integration Server は、キャッシュされたエレメントごとに、キャッシュされたエレメントのキー、キャッシュされたサービス入力、およびそれに対応するキャッシュされたサービス出力を表示します。

41 保証付きデリバーの設定

- 保証付きデリバーとは 790
- 保証付きデリバーのためのサーバ設定 791
- 保証付きデリバーの管理 795

保証付きデリバーとは

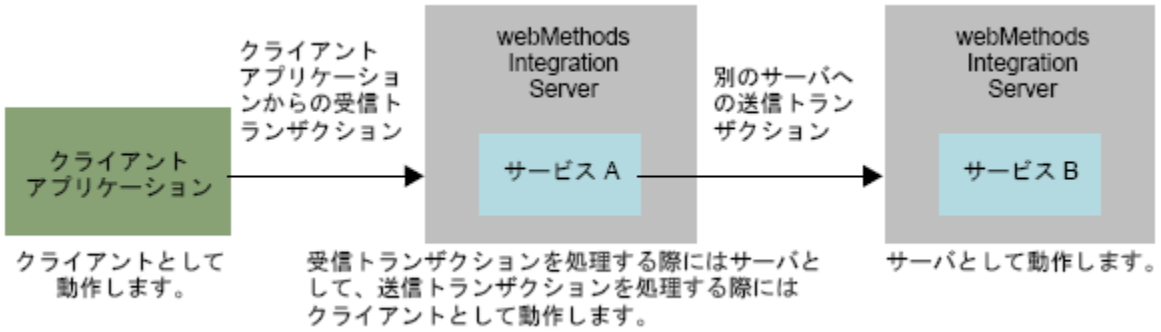
Integration Server は、1 回限りの保証付きサービスを確実に実行する保証付きデリバー機能を装備しています。

Integration Server の保証付きデリバー機能を使用すると、以下のタスクが確実に実行されます。

- クライアントからのサービス実行要求をサーバにデリバーします。
- サービスを 1 回だけ実行します。
- サービス実行からの応答をクライアントにデリバーします。

webMethods 保証付きデリバー機能は、ネットワーク、クライアント、またはサーバ上で発生する一時的な障害に対処します。一時的な障害とは、自動的に修復される障害です。一時的障害によってサーバに要求をデリバーできない場合は、要求が再送信され、障害が修復されたときに再試行されて要求が正常にデリバーされます。保証付きデリバートランザクションの TTL (time-to-live: 廃棄までの時間) のほかに、オプションとしてトランザクション再試行回数を指定することによって、一時的エラーとなる要件が決定されます。

Integration Server は、保証付きデリバートランザクションにおいてサーバまたはクライアントのいずれとしても機能するため、サーバの保証付きデリバー機能は受信トランザクションと送信トランザクションの両方を処理します。クライアントがサーバ上のサービスを呼び出す場合には、そのサーバはサーバとして機能します。サービスが保証付きデリバーを使用して別の Integration Server 上のサービスを呼び出す場合には、サービスを呼び出したサーバはクライアントになります。



注:
保証付きデリバートランザクションに参加する Integration Server は、どちらも同じバージョンの Integration Server ソフトウェアを実行している必要があります。

保証付きデリバー機能を使用すると、一時的障害に対応するための複雑なエラー処理コードを組み込まなくても、堅牢なトランザクションベースのクライアントアプリケーションを作成できます。

重要:
ある要求から次の要求へ状態情報を維持することはできないので、ステートレスな (アトミック) トランザクションで保証付きデリバー機能を使用します。このため、保証付きデリバー機能は、複数の要求による対話型サービスには使用できません。

保証付きデリバーの詳細については、『*Guaranteed Delivery Developer's Guide*』を参照してください。

保証付きデリバーのためのサーバ設定

Integration Server では、保証付きデリバートランザクションに対してさまざまな設定を使用します。設定の大部分にはデフォルトがあります。通常はデフォルトを使用します。設定の変更には、146 ページの「拡張設定の使い方」に記載されているように、Integration Server Administrator の [**設定**] > [拡張設定] ページを使用します。

保証付きデリバーが機能するには、ISInternal 機能エイリアス ([**設定**] > [JDBC プール] ページで指定) を組み込み IS Internal データベースまたは外部 RDBMS のいずれかをポイントするように設定する必要があります。データベースコンポーネントへの Integration Server の接続の詳細については、157 ページの「Integration Server を外部 RDBMS のデータベースコンポーネントへ接続する」を参照してください。

サーバクラスタリングで保証付きデリバーを使用する場合の詳細については、『*webMethods Integration Server Clustering Guide*』を参照してください。

受信トランザクションと送信トランザクションで共有する設定

受信保証付きデリバートランザクションと送信保証付きデリバートランザクションの両方で、次の設定を使用します。

watt.server.txMail
watt.server.txMail 設定は、保証付きデリバー機能がエラーのため無効となった場合 (サーバがディスクフルの状態になるなど) に、通知する管理者の電子メールアドレスを指定するために使用します。watt.server.txMail の設定例として、「watt.server.txMail=ISAdmin@YourCompany.com」があります。

watt.server.txMail の設定にデフォルトはありません。

watt.server.smtpServer
watt.server.smtpServer 設定は、Integration Server が保証付きデリバーのエラーについての電子メールメッセージを送信するときに使う SMTP サーバのドメイン名 (purple.webmethods.com など) または IP アドレス (132.196.19.22 または 2001:db8:85a3:8d3:1319:8a2e:370:7348 など) を指定するために使用します。watt.server.smtpServer の設定例として、「watt.server.smtpServer=132.196.19.22」や「watt.server.smtpServer=2001:db8:85a3:8d3:1319:8a2e:370:7348」があります。

watt.server.txMail の設定にデフォルトはありません。

管理者がエラー通知の電子メールを受信したら、問題の解決後に Integration Server Administrator を使用して保証付きデリバー機能を再初期化する必要があります。保証付きデリバーの再初期化の手順については、796 ページの「保証付きデリバーの再初期化」を参照してください。

watt.tx.vm.id
watt.tx.vm.idwatt.tx.vm.id 設定は、Integration Server ID を指定するために使用します。同じホストマシン上で複数の Integration Server が実行されていて、すべてのサーバで保証付きデリバーが使用されている場合に、このパラメータを使用します。各 Integration Server に一意

の ID を指定することによって、重複する保証付きデリバートランザクション ID が作成されるのを防止できます。値は 0～2147483647 の整数にする必要があります。

デフォルトは *0* です。

受信トランザクションの設定

受信トランザクションでは、トランザクションの「ジョブストア」とそれぞれの状態がサーバ上に保持されます。サーバは定期的にジョブストアをスイープし、期限切れトランザクション、つまり TTL を経過したトランザクションを削除します。受信要求の場合、クライアントがトランザクションに TTL を指定する必要があります。

ジョブストアに加えて、サーバは受信トランザクションに対応して実行したすべての操作の監査トレールログも管理します。

構成可能な受信トランザクション設定は、以下の表のとおりです。

設定内容	使用する設定
期限切れトランザクションを削除するためにジョブストアをスイープする頻度	watt.server.tx.sweepTime
ヒューリスティックな障害が発生した場合、障害の修復後に「ペンディング (Pending)」のトランザクションの状態を更新するかどうか	watt.server.tx.heuristicFailRetry

watt.server.tx.sweepTime
watt.server.tx.sweepTime 設定は、受信トランザクションのジョブストアのスイープ (消去) 間隔を秒数で指定するために使用します。サーバは、期限切れトランザクションを削除するためにジョブストアをスイープします。

デフォルトは *60 秒*です。

watt.server.tx.heuristicFailRetry
watt.server.tx.heuristicFailRetry 設定は、サーバが障害発生後に再起動した際、ペンディング (Pending) になっているジョブストア内のトランザクションに対して、サーバがサービスを再実行するかどうかを示します。トランザクションが「ペンディング (Pending)」になっている場合は、サービスが開始された後、サーバに障害が発生したときに実行が中断されたことを意味します。

サーバがサービス要求の正確な状態を特定できないため、サーバは保証付きトランザクションにヒューリスティックな障害が発生したと見なします。ヒューリスティックな障害に対して適切に対応するようにサーバを設定することもできます。デフォルトの watt.tx.heuristicFailRetry 設定では、ヒューリスティックな障害の後に引き続きサービスが再実行される可能性がありますが、サーバはサービスを少なくとも一回実行します。または、ヒューリスティックな障害が原因で再実行しないリスクがありますが、サービスが一回だけ実行されるよう設定しなおすことができます。

watt.tx.heuristicFailRetry 設定を「`true`」にすると、サーバはトランザクションの状態を「ペンディング (Pending)」から「新規 (New)」にリセットしてサービスを再試行します。設定が「true」

である場合、サーバがサービスを実行する前にトランザクションの期限が切れた場合にのみ、サービスの実行要求が失敗します(クライアントは、トランザクションが期限切れになる時期を指定します)。

watt.tx.heuristicFailRetry 設定を「false」にすると、サーバはトランザクション状態を「ペンディング (Pending)」から「失敗 (Fail)」にリセットしてヒューリスティックな障害が発生したことを示し、サーバはサービスを再試行しません。設定が「false」である場合は、ヒューリスティックな障害が発生したとき、またはトランザクションの期限が切れたときにサービスの実行要求が失敗します。

デフォルトは「*true*」です。

送信トランザクションの設定

送信トランザクションに保証付きデリバーを使用しないようにすることができます。ただし、送信トランザクションに保証付きデリバーを許可すると、サーバはこれらのトランザクションのジョブストアを個別に管理します。受信ジョブストアと同様に、サーバは送信ジョブストア内の各トランザクションの状態を維持します。サービス要求が失敗すると、サーバは要求を再サブミットする前に指定時間待機します。サーバは、サブミットが必要なトランザクションを識別するためにジョブストアを定期的に処理します。

サーバは、ペンディングの送信要求を処理するためのスレッドプールを維持しています。サーバがスレッドプール内に維持する必要があるクライアントスレッドの数を設定できます。

また、サーバは送信トランザクションに実施したすべての操作ごとに、監査トレールログを維持します。

構成可能な送信トランザクション設定は、以下の表のとおりです。

設定内容	使用する設定
送信トランザクションに保証付きデリバーを使用するかどうか	watt.tx.disabled
送信トランザクションのデフォルト TTL 値	watt.tx.defaultTTLMins
失敗した要求を再サブミットするまでサーバが待機する期間	watt.tx.retryBackoff
サブミットが必要なトランザクションを識別するためにサーバがジョブストアを処理する頻度	watt.tx.sweepTime
ペンディング要求を処理するために使用するスレッドプール内に維持するクライアントスレッドの数	watt.tx.jobThreads

watt.tx.disabled

watt.tx.disabled 設定を使用して、送信要求に対して保証付きデリバーの使用を無効にするように指定します。デフォルトで、サーバは送信トランザクションに対して保証付きデリバーの使用を許可します。デフォルトは「false」です。予期しない例外条件が発生した場合は、サーバによって保証付きデリバーが無効化されることがあります。この場合、watt.tx.disabled プロパティは「true」に設定されます。

デフォルトは「*false*」です。

watt.tx.defaultTTLMins
watt.tx.defaultTTLMins 設定を使用して、送信する保証付きデリバートランザクションに対してデフォルトの TTL (time-to-live: 廃棄までの時間) の値を指定します。送信トランザクションを開始するサービスが TTL 値を指定しない場合に、サーバが送信トランザクションをジョブストア内で管理する時間を分で指定します。

デフォルトは *30* です。

watt.tx.retryBackoff
watt.tx.retryBackoff 設定を使用して、サービス要求が失敗した後に、Job Manager が Integration Server へサービス実行要求を再サブミットするまで待機する秒数を指定します。

デフォルトは *60* です。

watt.tx.sweepTime
watt.tx.sweepTime 設定を使用して、送信トランザクションのジョブストアのスイープ間隔を秒数で指定するために使用します。サーバは、送信が必要なトランザクションを識別するためにジョブストアをスイープします。

デフォルトは *60* です。

watt.tx.jobThreads
watt.tx.jobThreads 設定を使用して、ペンディング要求を処理するためのスレッドプール内で使用可能にするクライアントスレッドの数を指定します。

デフォルトは *5* です。

エラーメッセージ用の電子メールアドレスと SMTP サーバの指定

保証付きデリバーを設定するには、保証付きデリバーが使用できなくなった場合に Integration Server がエラーメッセージを発行する宛先となる電子メールアドレスを指定する必要があります。さらに、電子メールメッセージを処理する SMTP サーバのドメイン名または IP アドレスを指定する必要があります。電子メールアドレスと SMTP サーバの指定方法については、241 ページの「重大な問題に関するメッセージの電子メールアドレスへの送信」を参照してください。

ISInternal データベースを共有する複数のサーバでの保証付きデリバーの使用

Integration Server クラスタに含まれない複数の Integration Server を使用するが、それらのサーバに、同じデータベースを参照する ISInternal 機能エイリアスが設定されている場合は、保証付きデリバージョブを正しく処理するように各 Integration Server を設定する必要があります。

ISInternal データベースを共有する各サーバでは、watt.server.db.share.ISInternal サーバプロパティを「`true`」に設定します。この設定により、サーバはデータベースを共有していること、および他のサーバと連携する必要があることを認識します。

また、サーバは、1 つの Integration Server クラスタに含まれていない場合でも、保証付きデリバーのテーブルを共有しています。このため、すべてのサーバで、以下のクラスタサーバプロパティの設定を同じにする必要があります。

- watt.server.tx.cluster.lockTimeoutMillis
- watt.server.tx.cluster.lockBreakSecs
- watt.server.tx.cluster.jobPendingWait

サーバプロパティの設定を表示または変更するには、146 ページの「拡張設定の使い方」の説明に従って、Integration Server Administrator で [**設定**] > [拡張設定] ページを使用します。サーバプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

重要:
これらのサーバプロパティを変更した後、変更内容を反映するには Integration Server を再起動する必要があります。

保証付きデリバーの管理

サーバを初期化すると、サーバは保証付きデリバー機能を初期化します。保証付きデリバーをシャットダウン、再初期化、およびテストするには、Integration Server Administrator を使用します。

保証付きデリバーのシャットダウン

保証付きデリバーサーバをシャットダウンすることなく、保証付きデリバー機能をシャットダウンおよび再度有効化できます。

設定エラーを修正したり新しい監査トレールログを開始したりする管理機能を実行するために、保証付きデリバーをシャットダウンする場合があります(新しい監査トレールログを開始するには、既存のログを移動または名称変更します。既存のログがない場合、サーバは自動的に新規のログを開始します)。

保証付きデリバーをシャットダウンするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. パッケージのリストで [**WmPublic**] をクリックします。
4. [**WmPublic 内のサービスを表示**] をクリックします。
5. サービスのリストで、[**pub.tx:shutdown**] をクリックします。
6. [**shutdown のテスト**] をクリックします。サーバにサービスのテストページが表示されます。
7. [**テスト (入力値なし)**] をクリックします。サーバによって、受信トランザクションへの保証付きデリバー機能が無効になります。

保証付きデリバーの再初期化

保証付きデリバーが無効になっている場合は、保証付きデリバーを再初期化します。保証付きデリバーを再初期化する手順は、受信トランザクションと送信トランザクションとで異なります。

受信トランザクションへの保証付きデリバーの再初期化

設定の問題を解決したり管理上の変更を加えたりするために保証付きデリバー機能をシャットダウンする場合は、Integration Server Administrator を使用して保証付きデリバーを再初期化します。

この手順を使用すると、エラー (ディスクフルの状態になる、ジョブストアデータベースにアクセスできなくなるなど) のために保証付きデリバーが使用できなくなった場合、保証付きデリバーを再初期化することもできます。問題を解決してから保証付きデリバーを再初期化します。

受信トランザクションへの保証付きデリバーを再初期化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. パッケージのリストで [**WmPublic**] をクリックします。
4. [**WmPublic 内のサービスを表示**] をクリックします。
5. サービスのリストで、[**pub.tx:init**] をクリックします。
6. [**init のテスト**] をクリックします。サーバにサービスのテストページが表示されます。
7. [**テスト (入力値なし)**] をクリックします。サーバによって、受信トランザクションへの保証付きデリバー機能が再初期化されます。

送信トランザクションへの保証付きデリバーの再初期化

送信トランザクションへの保証付きデリバー機能が、エラー (ディスクフルの状態になったり、ジョブストアデータベースにアクセスできないなど) のため使用できなくなった場合は、問題を解決してからこの手順を使用して保証付きデリバーを再初期化します。

送信トランザクションへの保証付きデリバーを再初期化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. パッケージのリストで [**WmPublic**] をクリックします。

4. [**WmPublic 内のサービスを表示**] をクリックします。

5. サービスのリストで、[**pub.tx:resetOutbound**] をクリックします。

6. [**resetOutbound のテスト**] をクリックします。サーバにサービスのテストページが表示されます。

7. [**テスト (入力値なし)**] をクリックします。サーバによって、送信トランザクションへの保証付きデリバー機能が再初期化されます。

42 Integration Server での Terracotta Ehcache の設定

- Terracotta Ehcache とは 800
- キャッシュの設定 800
- キャッシュとキャッシュマネージャについて 804
- キャッシュマネージャの設定ファイル 805
- Terracotta ライセンスのインストール、表示および変更 807
- オンヒープキャッシュの設定 809
- BigMemory キャッシュの設定 811
- 分散キャッシュの設定 814
- キャッシュを検索可能にする 823
- キャッシュマネージャの使用 826
- キャッシュの使用 835
- Terracotta Ehcache アクティビティのログへの記録 849

Terracotta Ehcache とは

Terracotta Ehcache は、Integration Server で使用される標準ベースのキャッシュ API です。キャッシュを使用すると、アプリケーションで頻繁に使用されるデータをメモリ (または他の周辺リソース) からフェッチできます。データが必要になるたびにデータベースや他のバックエンドシステムから抽出しなくても済みます。

Terracotta Ehcache は、複数の展開設定をサポートするように設計されています。展開設定には、非常に大きなメモリ内キャッシュを作成して複数の Integration Server で共有できるようにする設定などがあります。

Integration Server および Integration Server で実行される Software AG コンポーネントでは、Terracotta Ehcache を使用して、独自の内部処理に関連付けられているデータをキャッシュします。また、Integration Server では、開発者が構築するソリューションにキャッシュ機能を追加するために使用できるパブリックサービスのセット (WmPublic パッケージの pub.cache フォルダ内) が提供されています。たとえば、販売カタログからデータを頻繁に読み取るサービスでは、そのカタログのすべてまたは一部をキャッシュすることにより、カタログアイテムに関する情報が必要になるたびにデータベースをクエリーすることを回避できます。メモリからのデータへのアクセスはデータベースからのデータの抽出より 1,000 倍高速であるため、開発者はサービスで使用されるデータをキャッシュすることにより、サービスのパフォーマンスを劇的に改善することができます。

この章では、Terracotta Ehcache により提供されるキャッシュ機能の概要について説明します。これらの機能の詳細な説明については、Terracotta Ehcache の製品マニュアルを参照してください。

キャッシュの設定

Integration Server のキャッシュは、Java 仮想マシン (JVM) 内のヒープの一部を占有します。キャッシュのこの部分はオンヒープキャッシュと呼ばれます。 オプションで、キャッシュを一般的に「層」と呼ばれる以下の場所にヒープを越えて拡張することができます。

- ローカルディスク記憶領域
- BigMemory (Terracotta ライセンスが必要です)
- Terracotta Server Array (Terracotta ライセンスが必要です)

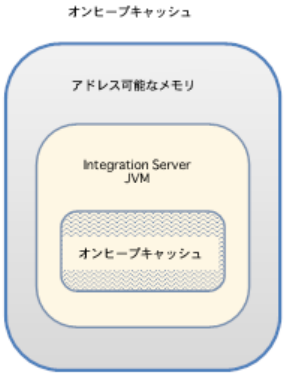
キャッシュをオンヒープメモリと他の層に分割する方法は、キャッシュを作成するときに指定する設定プロパティによって決まります。

層構造のアプローチでは、ヒープにより制限されるサイズの制約を超えてキャッシュを拡張することができます。また、キャッシュを Terracotta Server Array に拡張すると、複数の Integration Server でキャッシュを同時に共有できます。

Terracotta Ehcache を使用するサービスを開発する開発者は、使用するキャッシュがヒープにのみ格納されているのか、他のいずれかの層に拡張されているのかを把握しておく必要はありません。特定のキャッシュを使用するようにサービスをコーディングするだけです。このキャッシュがディスク、BigMemory または Terracotta Server Array に拡張されるかどうかは、キャッシュ自体に割り当てられている設定プロパティによって決まります。このため、アプリケーションのキャッシュ要件が変更された場合でも、開発者はコードを変更する必要はありません。

オンヒープキャッシュ

オンヒープキャッシュは、キャッシュのうち、Integration Server が稼動している JVM のヒープ内に格納される部分を指します。



オンヒープキャッシュは高速ですが、ヒープ領域に格納されるため、ガーベッジコレクションプロセスの対象となります。キャッシュのサイズが大きい場合は、サービスの実行中にガーベッジコレクションプロセスにより長い中断が生じる場合があります。ヒープが大きい場合、中断が数秒続くことがあります。

注:
オンヒープキャッシュに使用できるメモリ量は、JVM ヒープに割り当て可能なメモリ量によって異なります。この制限は、使用する JVM、マシンおよびオペレーティングシステムによって異なります。

Integration Server では、デフォルトでオンヒープキャッシュの使用がサポートされています。オンヒープキャッシュの作成や使用には、追加のライセンスは必要ありません。

Integration Server でのオンヒープキャッシュの使用の詳細については、809 ページの「オンヒープキャッシュの設定」を参照してください。

ローカルディスクストア

オプションで、ローカルディスクストアをキャッシュと関連付けることができます。ローカルディスクストアとは、Integration Server がキャッシュに配置されたオブジェクトを書き込むためのディレクトリのことです。以下の 2 つの方法のいずれかに従って、ローカルディスクストアを使用するようにキャッシュを設定できます。

- ローカルディスクストアを使用してオンヒープキャッシュを拡張し、オンヒープキャッシュがいっぱいになったときに、キャッシュされたオブジェクトがローカルディスクストアにオーバーフローするように設定できます。

  または

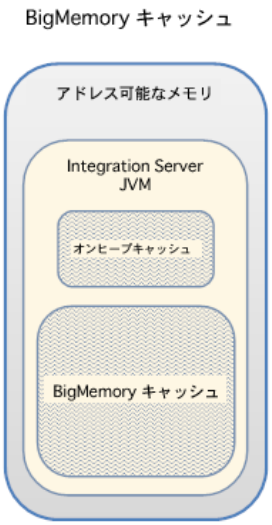
- ローカルディスクストアを使用して、キャッシュされたすべてのオブジェクトをディスクに保存することができます。これにより、Integration Server を再起動したり、キャッシュマネージャを再初期化したりするときに、キャッシュされたオブジェクトが保持されます。

  注:
  キャッシュをローカルディスクストアにパーシストしても、Integration Server またはキャッシュマネージャの再起動時にキャッシュ内のエレメントが正常に回復することは保証されません。Integration Server またはキャッシュマネージャが正常にシャットダウンしない場合は、ディスクストアがエレメントを回復できる状態にならないことがあります。キャッシュのパーシスタンスに保証が必要な場合は、Terracotta Server Array を使用する必要があります。

注:
ローカルディスクストアは、ローカルキャッシュと共にのみ使用できます。 ローカルキャッシュは、Integration Server が稼動しているマシンに全体的に格納される Terracotta Ehcache ベースのキャッシュであり、Terracotta Server Array には拡張されません。

BigMemory

BigMemory を使用すると、キャッシュをローカル JVM ヒープを超えて拡張できます。BigMemory を使用するようにキャッシュを設定すると、キャッシュの一部がヒープ内に格納され、別の部分がオフヒープに格納されます。キャッシュのオフヒープ部分は JVM プロセスメモリ内に存在していますが、JVM ヒープの外部に格納されることになります。



BigMemory キャッシュはオフヒープに格納されるため、JVM ガーベッジコレクションプロセスの対象にはならず、このため予想どおりに一貫して実行されるようになります。キャッシュのサイズが大きい場合は、一般に BigMemory のパフォーマンスはヒープベースのキャッシュよりも向上します。さらに、管理の観点では、BigMemory を使用するキャッシュはガーベッジコレクション用に調整する必要がないため管理が容易です。

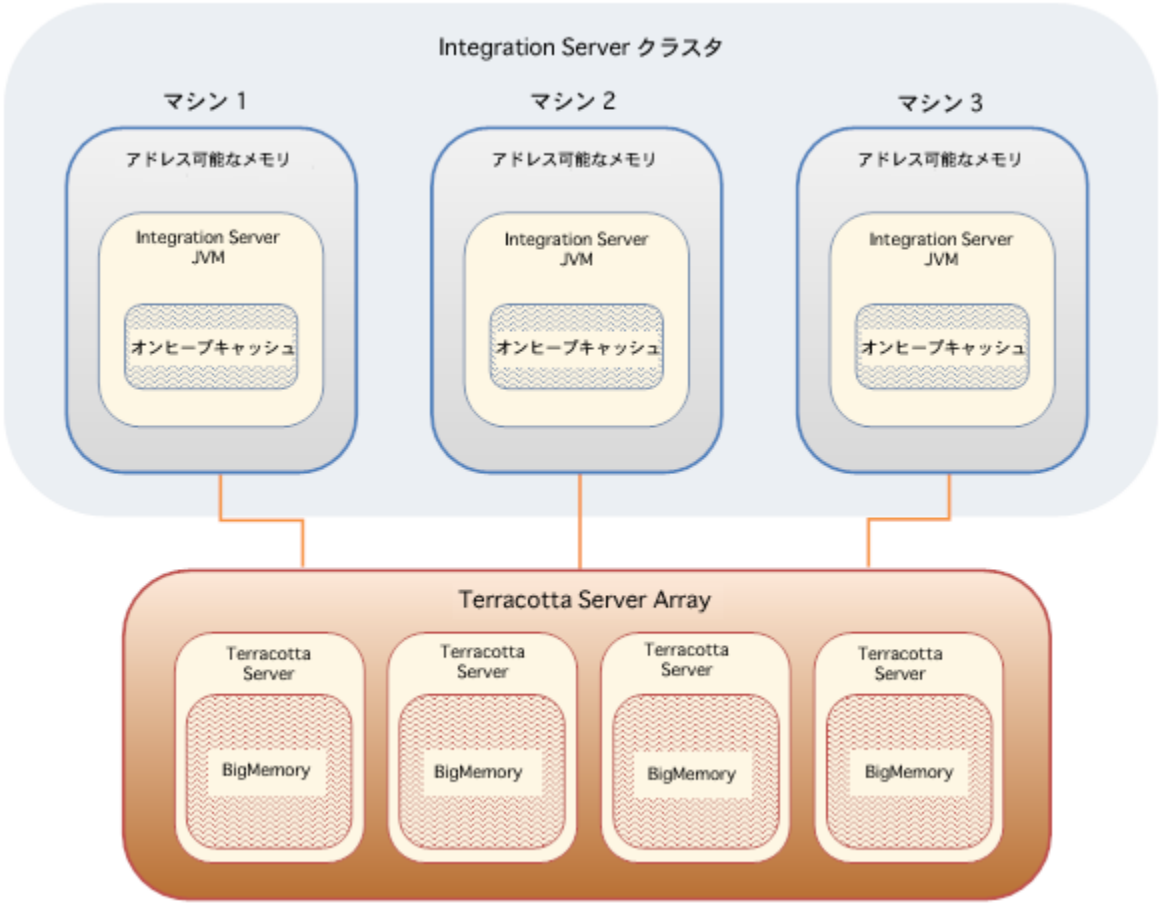
BigMemory を使用すると、オンヒープメモリ単独の場合よりも大きなキャッシュを作成できます。ヒープサイズの制限によって制約されるオンヒープキャッシュとは異なり、BigMemory を使用すると最大で 1 テラバイトのメモリをキャッシュに使用できます。

BigMemory を使用するには、Integration Server に Software AG から Terracotta ライセンスを取得している必要があります。

BigMemory を使用するためのキャッシュの設定の詳細については、811 ページの「BigMemory キャッシュの設定」を参照してください。

Terracotta Server Array

Terracotta Server Array を使用すると、分散キャッシュを作成できます。分散キャッシュは、複数の Integration Server で共有できます。たとえば、クラスタ化された Integration Server は、分散キャッシュを使用して、クラスタ内のすべての Integration Server が共有する必要のあるデータを保持します。



分散キャッシュを作成すると、キャッシュの完全なコピーが Terracotta Server Array に格納されます。Terracotta Server Array は 1 つ以上の Terracotta Server で構成されており、大量のデータ (数テラバイト) をキャッシュできます。キャッシュのデータは、ストライピングと呼ばれる技術を使用して Terracotta Server に分散されます。Integration Server 上の Terracotta クライアントにより、Integration Server と Terracotta Server Array 上のキャッシュの間のインタラクションが管理されます。

キャッシュを共有する Integration Server は Terracotta Server Array に接続し、キャッシュにデータを書き込み、キャッシュからデータを抽出します。各 Integration Server は、最近使用したデータのホットセットとして機能するキャッシュの一部をローカルに保持し、Terracotta Server Array とのやり取りの回数を減らします。

分散キャッシュを使用するには、Integration Server に Software AG から Terracotta ライセンスを取得している必要があります。

Integration Server での分散キャッシュの使用の詳細については、814 ページの「分散キャッシュの設定」を参照してください。

キャッシュとキャッシュマネージャについて

Terracotta Ehcache では、キャッシュはキーと値のペアとして表されるエレメントを保持します。キーとその値はどちらも Java オブジェクトです。キャッシュ内のキーと値のペアに関して、以下の点に留意してください。

- キーは、IS ドキュメントを除く、任意のタイプの Java オブジェクトにできます。
- 値は、任意のタイプの Java オブジェクトにできます。
- データを BigMemory または Terracotta Server Array にキャッシュするか、またはローカルディスクストアにキャッシュを書き込む場合、オブジェクトはシリアライズ可能である必要があります。

  注:
  IData インプリメンタはシリアライズ可能であるため、キャッシュに値として含めることができます。ただし、IData 自体にシリアライズ可能オブジェクトを含める必要があります。キャッシュ対象の IData にストリーム、XML ノード、シリアライズ不能なカスタムオブジェクトなど、シリアライズできないオブジェクトが含まれている場合、Terracotta はキャッシュへの書き込み試行中に例外をスローします。

キャッシュに配置されたエレメントは、キーで識別されます。たとえば、Integration Server がセッションオブジェクトをキャッシュするときは、キーとしてセッション ID を使用します。同様に、開発者がキャッシュにオブジェクトを書き込むサービスを作成するときは、適切な識別子を選択してキーとして使用し、キャッシュからオブジェクトを抽出します。たとえば、顧客オブジェクトをキャッシュするアプリケーションでは、顧客のアカウント番号からキーを導出できます。

キャッシュにはキャッシュマネージャが関連付けられています。 キャッシュマネージャは、キャッシュのセットをまとめて開始したりシャットダウンしたりできる 1 つの管理単位にグループ化するコンテナとして機能します。

キャッシュマネージャと関連付けるキャッシュには、異なる特性や設定を指定できます。たとえば、キャッシュマネージャ XYZ には、全体がヒープに格納されるキャッシュと、BigMemory に拡張される別のキャッシュを含めることができます。

多くのプロパティはキャッシュベースでキャッシュに設定することができますが、特定のプロパティはキャッシュマネージャレベルで割り当てます。以下に例を示します。

- **ローカルディスクストアの場所**。1 つのキャッシュマネージャに関連付けられているすべてのキャッシュは、同じディスクストアの場所を使用します。2 つのキャッシュで異なるロー

カルディスクストアを使用する場合は、異なるキャッシュマネージャにこれらのキャッシュを作成する必要があります。

- **Terracotta Server Array の場所**。1 つのキャッシュマネージャに関連付けられているすべての分散キャッシュは、同じ Terracotta Server Array を使用します。2 つのキャッシュで 2 つの異なる Terracotta Server Array を使用する場合は、異なるキャッシュマネージャにこれらのキャッシュを作成する必要があります。

Integration Server で作成するキャッシュマネージャは、パブリックキャッシュマネージャと呼ばれます。パブリックキャッシュマネージャ (およびパブリックキャッシュマネージャに含まれるキャッシュ) は、開発者が構築するフローサービスで使用可能です。

パブリックキャッシュマネージャとキャッシュの作成および設定の詳細については、826 ページの「キャッシュマネージャの使用」および835 ページの「キャッシュの使用」を参照してください。

システムキャッシュ

Integration Server およびその他の Software AG 製品では、それぞれの内部処理の多くで Terracotta Ehcache を使用します。使用されるキャッシュはシステムキャッシュと呼ばれ、システムキャッシュマネージャに属しています。

システムキャッシュマネージャは、次の文字で始まる名前で識別されます。

```
SoftwareAG
```

たとえば、次のように入力します。

```
SoftwareAG.IS.Core
```

Integration Server Administrator を使用して Integration Server に存在するキャッシュマネージャのリストを表示すると、システムキャッシュマネージャとパブリック (ユーザ定義) キャッシュマネージャが別のリストに表示されます。

Integration Server で使用されるシステムキャッシュの一覧については、『*Using BigMemory with webMethods Products*』を参照してください。このマニュアルでは、Integration Server およびその他の Software AG コンポーネントが使用するシステムキャッシュに関する説明と、システムキャッシュのサイズ設定情報が提供されています。

キャッシュマネージャの設定ファイル

各キャッシュマネージャは、キャッシュマネージャとそのキャッシュの設定パラメータを指定する XML ファイルに関連付けられています(Terracotta Ehcache のマニュアルでは、多くの場合このファイルは ehcache 設定ファイルまたは ehcache.xml ファイルと呼ばれます)。

Integration Server は、すべてのキャッシュマネージャ (システムキャッシュマネージャおよびパブリックキャッシュマネージャ) の設定ファイルを次のディレクトリで管理します。

Integration Server_directory \instances*instance_name*\config\Caching

Integration Server Administrator を使用して新しいキャッシュマネージャを作成すると、Integration Server はキャッシュマネージャの設定ファイルを作成し、このファイルを

Integration Server_directory \instances*instance_name*\config\Caching ディレクトリに保存します。Integration Server は、設定ファイルにキャッシュマネージャと同じ名前を付けます。ただし、名前に含まれるピリオドの代わりにダッシュ (-) が使用されます。たとえば、`my.cache.manager` という名前のキャッシュマネージャの設定ファイルには、`my-cache-manager.xml` という名前が付けられます。

Integration Server を起動すると、 *Integration Server_directory* \instances*instance_name*\config\Caching ディレクトリ内の設定ファイルが検索されます。有効な設定ファイルを見つけると、そのファイルで定義されているキャッシュマネージャを初期化し、そのキャッシュマネージャに属するキャッシュを開始します。

Integration Server の起動プロセス中にキャッシュマネージャを初期化できない場合 (キャッシュマネージャの設定ファイルが無効であったり、ライセンスで使用が許可されていないタイプのキャッシュが設定ファイルで定義されている場合など)、サーバログにエラーがレポートされます。

注:
開発者は、Terracotta Ehcache API を使用すると、対応する設定ファイルを *Integration Server_directory* \instances*instance_name*\config\Caching ディレクトリに追加することなく、キャッシュマネージャおよびキャッシュを作成できます。ただし、この方法は Integration Server ではお勧めしません。設定ファイルが *Integration Server_directory* \instances*instance_name*\config\Caching ディレクトリに存在しないキャッシュマネージャは Integration Server に正常に登録されず、Integration Server Administrator を使用して管理することはできません。

キャッシュのパラメータの指定

キャッシュを作成し、そのパラメータを編集するには、Integration Server Administrator を使用します。キャッシュを作成すると、そのキャッシュは、Integration Server Administrator によって適切なキャッシュマネージャの設定ファイルに追加されます。キャッシュのパラメータを編集すると、それに応じて Integration Server Administrator によってキャッシュマネージャの設定ファイル内のパラメータが更新されます。

重要:
キャッシュマネージャの設定ファイルを手動で編集してキャッシュのパラメータを変更できますが、この方法はお勧めしません。キャッシュマネージャまたはキャッシュのパラメータを編集する場合は、常に Integration Server Administrator を使用してください。設定ファイルを手動で編集する必要があるのは、Integration Server Administrator ユーザインタフェースに表示されないパラメータを設定する場合のみです (ほとんど必要ありません)。設定ファイルの編集に関するその他の注意については、833 ページの「キャッシュマネージャの設定ファイルの手動編集」を参照してください。

動的および非動的キャッシュパラメータ

キャッシュには、動的パラメータと非動的パラメータがあります。 Integration Server Administrator を使用して動的パラメータを変更すると、変更内容は直ちにキャッシュに適用されます。変更内容を有効にするために Integration Server を再起動したりキャッシュマネージャを再初期化したりする必要はありません。非動的パラメータを変更すると、831 ページの「キャッ

シュマネージャの再初期化」で説明されているようにキャッシュマネージャを再初期化するまで、変更内容は有効になりません。

835ページの「キャッシュの作成」のパラメータの説明に、どのパラメータが動的でどのパラメータが動的でないかが示されています。

注:
動的であるか非動的であるかとは別に、分散キャッシュのパラメータはキャッシュワイドまたはクライアント固有でもあります。 これらの性質の詳細については、818 ページの「分散キャッシュのキャッシュワイドのパラメータとクライアント固有のパラメータ」を参照してください。

Terracotta ライセンスのインストール、表示および変更

Terracotta コンポーネントのライセンス要件は、使用するコンポーネントのタイプによって異なります。ローカルのオンヒープキャッシュを使用するには、追加の Terracotta ライセンスは必要ありません。この機能は Integration Server と共にインストールされる基本ライセンスで提供されます。ただし、BigMemory または Terracotta Server Array を使用するキャッシュを作成するには、適切な Terracotta ライセンスおよび Integration Server ライセンスが必要です。

次の表に、各 Terracotta コンポーネントのライセンス要件を示します。

使用するコンポーネント	必要なライセンス
ローカルキャッシュ	■ Integration Server ライセンス。このライセンスは Integration Server のインストールと共にインストールされます。 **注:** 追加の Terracotta ライセンスはこのコンポーネントには必要ありません。
BigMemory キャッシュ	■ Integration Server ライセンス。このライセンスファイルは Integration Server のインストールと共にインストールされます。 ■ BigMemory の Terracotta ライセンス。このライセンスを、Integration Server が稼動しているホストマシンに追加します。
分散キャッシュ	■ 分散キャッシュの Integration Server ライセンス。このライセンスは Integration Server のインストールと共にインストールされます。 ■ Terracotta Server Array のアップグレードされた Terracotta BigMemory ライセンス。このライセンスを、分散キャッシュを共有する各 Integration Server に追加します。

使用するコンポーネント	必要なライセンス
クラスタリング	■ クラスタリングの Integration Server ライセンス。このライセンスは Integration Server のインストールと共にインストールされます。 ■ Terracotta Server Array の Terracotta BigMemory ライセンス。このライセンスを、クラスタ内の各 Integration Server に追加します。
Terracotta Server Array とオフヒープ記憶領域とのクラスタリング	■ クラスタリングの Integration Server ライセンス。このライセンスは Integration Server のインストールと共にインストールされます。 ■ Terracotta Server Array のアップグレードされた Terracotta BigMemory ライセンス。このライセンスを、クラスタ内の各 Integration Server に追加します。

Terracotta ライセンスがあるかどうかの確認

Terracotta ライセンスがあるかどうかが不明な場合は、Integration Server Administrator の [**サーバ**] > [ライセンス] ページで、お使いのシステムでライセンスされているコンポーネントを表示できます。Terracotta ライセンスがインストールされている場合は、ここに表示されます。[**ライセンスの詳細**] をクリックすると、ライセンスで使用が許可されている Terracotta コンポーネントを表示できます。Terracotta ライセンスがない場合は、Software AG から取得できます。

Terracotta ライセンスの追加

Terracotta ライセンスは、terracotta-license.key という名前のファイルです。このファイルには、すべての Terracotta コンポーネントに関するライセンス情報が含まれています。このファイルを Integration Server に追加するには、Integration Server が稼動しているマシンの *Software AG_directory* \common\conf ディレクトリにこのファイルを配置します。Integration Server Administrator は、このディレクトリでライセンスをチェックし、[ライセンス] ページに情報を表示します。

ライセンスファイルが異なるディレクトリに配置されている場合は、Integration Server Administrator を使用してファイルを Integration Server に追加します。

Integration Server Administrator を使用して Terracotta ライセンスキーを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ライセンス] に移動します。
3. [**ライセンスの詳細**] をクリックします。
4. [**ライセンスの詳細の編集**] をクリックします。

5. [**Terracotta ライセンスファイル**] フィールドに、Software AG から取得したライセンスキーファイルの完全修飾名を入力します。

6. [**変更内容の保存**] をクリックします。Integration Server Administrator で、入力したファイル名を示すように [**Terracotta ライセンスファイル**] フィールドが更新されます。

7. Integration Server を再起動します。

注:
Terracotta ライセンスを追加、削除、または変更した後は、Integration Server を再起動する必要があります。

オンヒープキャッシュの設定

オンヒープキャッシュを作成するには、以下の一般的な手順に従います。オンヒープキャッシュはすべて JVM ヒープ内で管理されます。オンヒープキャッシュは高速ですが、JVM ガーベッジコレクションプロセスの対象になります。オンヒープキャッシュの詳細については、801 ページの「オンヒープキャッシュ」を参照してください。

手順	説明
1	810ページの「オンヒープキャッシュの設定に関する考慮事項」に記載されている内容を確認します。
2	キャッシュに必要なメモリ量を決定し、必要に応じて JVM のヒープサイズを調整します。Integration Server が使用するヒープサイズの変更の詳細については、149 ページの「Integration Server が使用する JVM ヒープサイズの変更」を参照してください。
3	キャッシュを追加するキャッシュマネージャがまだ存在しない場合は作成します。キャッシュマネージャの作成の詳細については、826 ページの「キャッシュマネージャの作成」を参照してください。
4	835 ページの「キャッシュの作成」の手順に従ってキャッシュを作成します。キャッシュを作成するときは、以下の作業を実行する必要があります。 ■ [**キャッシュ設定**] セクションおよび [**キャッシュの高度な設定**] セクションでパラメータを設定します。 ■ [**分散キャッシュの設定**] セクションの [**分散**] オプションを有効にしないでください。 ■ [**BigMemory**] セクションの [**オフヒープへオーバーフロー**] オプションを有効にしないでください。
5	キャッシュを使用できるようにするには、848ページの「キャッシュの有効化」の手順に従ってキャッシュを有効にします。

オンヒープキャッシュの設定に関する考慮事項

オンヒープキャッシュを作成するときは、以下の点に留意してください。

- ヒープ領域は、JVM で実行中のすべてのプロセスで共有されます。オンヒープキャッシュを Integration Server に追加するときに、新しいキャッシュや別のヒープコンシューマのヒープ要件を満たすのに十分な大きさになるよう、ヒープのサイズを増やす必要がある場合があります。以下に例を示します。
  - Integration Server に存在する他のオンヒープキャッシュ (システムキャッシュを含む)。
  - Integration Server で使用されるすべてのクラス、クラス変数、インスタンス変数、およびそれに展開されるサービスやトリガー。
  - Integration Server と同じ JVM で実行中の OSGI プロセス (webMethods Event Server など)。

  Integration Server およびその他の Software AG 製品で使用されるシステムキャッシュに関するサイズ設定については、Software AG マニュアルの Web サイトにある『*Using BigMemory with webMethods Products*』を参照してください。
- キャッシュにより消費されるヒープの量を推定するには、キャッシュするオブジェクトのサイズに、キャッシュに保存するオブジェクトの数を乗算します。

  objectSizeInBytes * *numObjectsToCache* = *heapConsumedByCache*
- オンヒープキャッシュのサイズが非常に大きいと (ヒープサイズを大きくする必要があります)、JVM でフルガーベッジコレクションを実行するためにかかる時間が増えます。フルガーベッジコレクションが発生すると、JVM のすべてのスレッドが一時停止し、応答時間が著しく遅延する場合があります。

  オンヒープキャッシュのサイズを設定するときは、大量のデータをメモリに保存する利点と、それによって Integration Server に生じる定期的な中断とを比較検討する必要があります。ガーベッジコレクションのキャッシュへの影響の詳細については、Ehcache product documentation for 2.8 at http://ehcache.org/documentationを参照してください。
- オンヒープキャッシュのサイズでガーベッジコレクションが重大な問題になる場合は、オンヒープキャッシュのサイズを小さくすること、および [**ディスクへオーバーフロー**] または [**オフヒープへオーバーフロー**] (BigMemory ライセンスが必要) を有効にしてオンヒープ要件を増やすことなくキャッシュを拡張することを検討してください。
- JVM のヒープ要件、およびマシンで実行される他のアプリケーションのメモリ要件をサポートするために、マシンに十分なメモリが搭載されていることを確認します。メモリが不十分な場合は、ページングが発生します。
- [**ディスクへオーバーフロー**] オプションまたは [**ディスクへ保存**] オプションを有効にする場合、キャッシュにはシリアライズ可能なキーと値のみが含まれている必要があります。サービスがシリアライズ可能ではないオブジェクトを、ディスクにオーバーフローまたはパーシストしたキャッシュに書き込もうとすると、サービスはランタイム例外を受け取ります。

BigMemory キャッシュの設定

BigMemory キャッシュを作成するには、以下の一般的な手順に従います。BigMemory キャッシュは、オンヒープメモリとオフヒープメモリの両方を占有します(このため、オフヒープキャッシュと呼ばれることもあります)。BigMemory キャッシュを設定するときに、Integration Server がオンヒープで管理するエレメントの数を指定します。また、キャッシュで使用可能なオフヒープメモリの量も指定できます。BigMemory キャッシュは高速で、サイズが非常に大きくなることがあります。キャッシュのオフヒープ部分は、JVM ガーベッジコレクションプロセスの対象にはなりません。BigMemory キャッシュの詳細については、802 ページの「BigMemory 」を参照してください。

BigMemory を使用するキャッシュを作成するには、以下の一般的な手順に従います。

手順	説明
1	Integration Server に BigMemory を使用するためのライセンスが取得されていることを確認し、ライセンスで許可されている BigMemory の量をチェックします。手順の詳細については、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。
2	まだ行っていない場合は、812 ページの「Integration Server へのダイレクトメモリ領域の割り当て 」の手順に従って Integration Server にダイレクトメモリを割り当てます。
3	813 ページの「BigMemory キャッシュの設定に関する考慮事項」に記載されている内容を確認します。
4	キャッシュのオンヒープ部分に必要なメモリの量を決定し、必要に応じて Java ヒープサイズを調整します。Integration Server が使用するヒープサイズの変更の詳細については、149 ページの「Integration Server が使用する JVM ヒープサイズの変更」を参照してください。
5	キャッシュを追加するキャッシュマネージャがまだ存在しない場合は作成します。キャッシュマネージャの作成の詳細については、826 ページの「キャッシュマネージャの作成」を参照してください。
6	835 ページの「キャッシュの作成」の手順に従ってキャッシュを作成します。キャッシュを作成するときは、以下の作業を実行する必要があります。 ■ [**メモリ内エレメントの最大数**] フィールドで、オンヒープメモリで管理するエレメント数を指定します(パフォーマンスの問題を避けるために、常に [**メモリ内エレメントの最大数**] を 100 エレメントより大きい値に設定してください)。 ■ [**オフヒープへオーバーフロー**] を有効にして、[**最大オフヒープ**] フィールドでこのキャッシュに割り当てるオフヒープメモリの量を指定します。
7	キャッシュを使用できるようにするには、848 ページの「キャッシュの有効化」の手順に従ってキャッシュを有効にします。

Integration Server へのダイレクトメモリ領域の割り当て

Integration Server で BigMemory を使用するには、Integration Server が稼動している JVM にダイレクトメモリ領域を割り当てる必要があります。

Integration Server に付与されるダイレクトメモリ領域の容量は、custom_wrapper.conf ファイルの `wrapper.java.additional.n=-XX:MaxDirectMemorySize` プロパティで決定されます。デフォルトでは、Integration Server のインストール時には `wrapper.java.additional.n=-XX:MaxDirectMemorySize` プロパティが設定されていません。

BigMemory を使用する場合は、`wrapper.java.additional.n=-XX:MaxDirectMemorySize` プロパティを、オフヒープキャッシュを保持するのに必要なメモリの量に設定する必要があります。

注:
Microservices Runtime,にダイレクトメモリ領域を割り当てるには、 *Integration Server_directory* /bin/setenv.bat(sh) の JAVA_CUSTOM_OPTs に `-XX:MaxDirectMemorySize` を追加します。Microservices Runtime への Java システムプロパティの受け渡しの詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

Integration Server にダイレクトメモリ領域を割り当てるには

1. Integration Server がインストールされているマシンで、次のフォルダに移動します。

   Software AG_directory \profiles\IS_*instance_name*\configuration

2. テキストエディタで、custom_wrapper.conf を開きます。

3. BigMemory が使用するすべてのキャッシュを反映するために必要な BigMemory の量を指定するには、`wrapper.java.additional.n=-XX:MaxDirectMemorySize` プロパティを追加します。(この量は、Terracotta ライセンス、システムの物理メモリの容量、およびマシン上で実行されているオペレーティングシステムや他のアプリケーションで必要となる物理メモリの容量により制限されます)。

   たとえば、500 メガバイトの BigMemory が必要であると判断し、Integration Server が Windows で稼動している場合は、custom_wrapper.conf ファイルの `wrapper.java.additional` プロパティを次のように追加します。

   ```
   wrapper.java.additional.n=-XX:MaxDirectMemorySize=500M
   ```

   n は、`wrapper.java.additional` プロパティのファイル中での次の未使用の通し番号です。

   注:
   このパラメータの値は、メガバイトには「m」または「M」を、ギガバイトには「g」または「G」を使用して表すことができます。

4. custom_wrapper.conf を保存して閉じます。

5. Integration Server を再起動します。

BigMemory キャッシュの設定に関する考慮事項

BigMemory を使用するキャッシュを作成する場合は、以下の点に留意してください。

- BigMemory キャッシュは、シリアライズ可能なキーと値のみを受け入れます。サービスがシリアライズ可能ではないオブジェクトを実行時にキャッシュに書き込もうとすると、サービスは例外を受け取ります。
- キャッシュのオンヒープ部分には、100以上のエレメントを割り当てます。オンヒープキャッシュのサイズが小さいと、効率的に実行されません。キャッシュのオンヒープ部分に割り当てられたエレメントが 100 未満の状態で BigMemory キャッシュが初期化されると、Integration Server により警告メッセージがログに記録されます。
- キャッシュのオフヒープ部分には、128 メガバイト以上を割り当てます。
- JVM に割り当てるダイレクトメモリのすべてがオフヒープキャッシュに使用できるわけではないことに注意してください。BigMemory がキャッシュ以外に使用する分として 32 メガバイトを予約する必要があります(つまり、オフヒープキャッシュに使用できるメモリの量は、*MaxDirectMemorySize* から 32 MB を引いた分になります)。
- 新しいキャッシュにオフヒープメモリを指定する前に、Integration Server に存在するオフヒープキャッシュによって既に割り当てられている BigMemory の量をチェックします。このキャッシュに指定するオフヒープメモリは、ダイレクトメモリの残量以内にする必要があります (前述のように、キャッシュ以外に使用する分として予約された 32 メガバイトがメモリから引かれます)。

  既存のキャッシュで既に使用されている BigMemory の量を特定するには、各キャッシュマネージャの [**設定**] > [**キャッシュ**] > [*CacheManagerName*] ページの [**BigMemory**] 列を確認します。
- Integration Server は、キャッシュをインスタンス化するときに、キャッシュのオフヒープ部分 ([**最大オフヒープ**] で指定したメモリの量) を割り当てます。[**最大オフヒープ**] で指定したメモリの量を取得できない場合、Integration Server はサーバログにエラーメッセージを書き込み、エラーログにスタックトレースを記録して、キャッシュを有効にしません。
- キャッシュに割り当てるオフヒープメモリには、実際にはオンヒープキャッシュにあるエレメントのコピーが含まれます。キャッシュに [**最大オフヒープ**] パラメータを指定するときは、この点に留意してください。たとえば、[**最大オフヒープ**] を 128 MB に設定し、エレメントがそれぞれ約 1 KB の場合、キャッシュではオフヒープメモリ内に約 128,000 のエレメントを保持できます。ただし、この数字にはオンヒープエレメントのコピーも含まれます。オンヒープキャッシュのサイズが 2,000 エレメントの場合、オフヒープキャッシュにはその 2,000 エレメントのコピーに加えて追加の 126,000 エレメントが保持されます。
- [**ディスクへオーバーフロー**] オプションを有効にし、[**ローカルディスクのエントリの最大数**] パラメータを使用してディスクに書き込み可能なエレメントの数を制限したい場合は、[**ローカルディスクのエントリの最大数**] を、[**最大オフヒープ**] のメモリ量で保持できるエレメントの数よりも大きい値に設定します。たとえば、[**最大オフヒープ**] に約 128,000 エレメントを保持し、追加で 50,000 エレメントをディスクにオーバーフローさせる場合は、[**ローカルディスクのエントリの最大数**] を 178,000 エレメントに設定します。

- BigMemory キャッシュに関連するその他の考慮事項については、『BigMemory Go product documentation for 4.1 at www.terracotta.org/documentation』を参照してください。

分散キャッシュの設定

分散キャッシュを作成するには、以下の一般的な手順に従います。分散キャッシュは Terracotta Server Array に格納され、他の Integration Server と共有できます。Integration Server は分散キャッシュの一部をオンヒープメモリに保持するため、キャッシュからデータをフェッチする必要があるたびに Terracotta Server Array にアクセスする必要はありません。分散キャッシュを設定するときに、Integration Server がキャッシュのオンヒープ部分でローカルに管理するエレメントの数を指定します。分散キャッシュの詳細については、803 ページの「Terracotta Server Array 」を参照してください。

手順	説明
1	Terracotta Server Array がまだインストールおよび設定されていない場合は、インストールと設定を行います。手順については、Terracotta Server Array の製品マニュアルを参照してください。 **注:** Terracotta Server Array では、Integration Server にインストールされている Terracotta のクライアントライブラリと互換性がある Terracotta のバージョンが実行されている必要があります。Terracotta Server Array のどのバージョンが Integration Server と互換性があるかを確認するには、Software AG マニュアルの Web サイト (http://documentation.softwareag.com) で『*BigMemory Compatibility with other Software AG Products*』を参照してください。
2	まだ行っていない場合は、Terracotta Server Array の tc-config ファイルを編集して Integration Server に必要なパラメータを追加します。手順の詳細については、816 ページの「Terracotta Server Array での tc-config.xml の設定 」を参照してください。
3	817 ページの「分散キャッシュの設定に関する考慮事項」に記載されている内容を確認します。
4	Integration Server に Terracotta Server Array を使用するためのライセンスが取得されていることを確認します。手順の詳細については、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。
5	Integration Server のキャッシュのオンヒープ部分に使用するメモリの量を決定し、必要に応じて Java ヒープサイズを調整します。Integration Server が使用するヒープサイズの変更の詳細については、149 ページの「Integration Server が使用する JVM ヒープサイズの変更」を参照してください。 **注:** 分散キャッシュのオンヒープ部分のサイズは、キャッシュを使用するすべての Integration Server で同じである必要があります。

手順	説明
6	キャッシュを追加するキャッシュマネージャがまだ存在しない場合は作成します。キャッシュマネージャの作成の詳細については、826 ページの「キャッシュマネージャの作成」を参照してください。 **注:** 使用するキャッシュマネージャが Terracotta Server Array を使用するように設定する必要があります。
7	835 ページの「キャッシュの作成」の手順に従ってキャッシュを作成します。キャッシュを作成するときは、以下の作業を実行する必要があります。 ■ ページの [分散キャッシュの設定] セクションの [**分散**] の設定を有効にします。 ■ [**メモリ内エレメントの最大数**] フィールドで、Integration Server のオンヒープメモリで管理するエレメント数を指定します。 **注:** Integration Server では [**メモリ内エレメントの最大数**] を 0 に設定できますが (これはエレメントの数がリソースの可用性によってのみ制限されることを示します)、Software AG は、上限を常に指定することを強くお勧めします。無制限のキャッシュは無制限に大きくなるため、マシンで使用できるリソースが枯渇した場合はパフォーマンスの問題またはシステムの障害が発生する可能性があります。 ■ [**キャッシュ内エントリの最大数**] フィールドでは、Terracotta Server Array 上で管理可能なキャッシュの最大サイズ (エレメントの数) を指定します。
8	キャッシュのパラメータが希望どおりに設定されていることを確認し、848 ページの「キャッシュの有効化」の手順に従ってキャッシュを有効にします。 これが分散キャッシュを作成した最初の Integration Server の場合は、Integration Server はこの手順の間に Terracotta Server Array で分散キャッシュを作成します。
9	他の Integration Server がこの分散キャッシュを使用する場合は、それぞれに対して手順 4～8 を繰り返します。 **注:** 手順 7 で他の Integration Server のそれぞれに新しいキャッシュを手動で指定する代わりに、キャッシュマネージャの設定ファイルをこの Integration Server から他の Integration Server にコピーすることにより、キャッシュを作成することもできます。そのためには、1) webMethods Deployer を使用してこの Integration Server から他の Integration Server にキャッシュマネージャを展開するか、2) キャッシュマネージャの設定ファイルをこの Integration Server から他の Integration Server に手動で

手順	説明
	コピーします。webMethods Deployer の使用の詳細については、『*webMethods Deployer User's Guide*』を参照してください。別の Integration Server にキャッシュマネージャの設定ファイルをコピーする方法の詳細については、831 ページの「Integration Server へのキャッシュマネージャの設定ファイルの追加 」を参照してください。

Terracotta Server Array での tc-config.xml の設定

Integration Server では、Terracotta Server Array を設定するために tc-config.xml ファイルでプロパティを設定する必要があります。必要なプロパティを設定して Integration Server を起動すると、Integration Server は tc-config.xml をダウンロードして Terracotta Server Array に接続するために必要な情報を取得します。

Integration Server Administrator の [**Terracotta Server Array URL**] パラメータを使用して、クラスタリングまたはキャッシュ設定のホストサーバを指定します。分散キャッシュを使用するパブリックキャッシュマネージャを開始するか、または分散システムキャッシュを使用するクラスタ化された Integration Server を起動すると、Integration Server は [**Terracotta Server Array URL**] パラメータを使用して、接続可能な最初に指定された Terracotta Server Array サーバから tc-config.xml をダウンロードします。tc-config.xml の設定を使用して、Integration Server は Terracotta Server Array に接続して分散キャッシュの使用を開始します。[**Terracotta Server Array URL**] パラメータの設定の詳細については、826 ページの「キャッシュマネージャの使用」 (分散キャッシュ) または『*webMethods Integration Server Clustering Guide*』(クラスタリングに Terracotta を使用する Integration Server) を参照してください。

tc-config.xml の設定は、Terracotta Server Array をインストールしてから行います。ただし、Integration Server で分散キャッシュを開始するか、または分散システムキャッシュを使用するクラスタ化された Integration Server を起動する前に行う必要があります。

tc-config.xml でパラメータを設定するには

1. Terracotta Server Array ホストサーバで、tc-config.xml をテキストエディタで開きます。

   注:
   デフォルトでは、Terracotta サーバは、*TerracottaHome*/server/bin フォルダの tc-config ファイルが見つかることを前提とします。tc-config.xml が Terracotta Server Array ホストサーバにまだ存在しない場合は、これを作成する必要があります。

2. 以下のエレメントコンテンツを指定します。

   注:
   以下のパラメータは、分散システムキャッシュを使用する Integration Server クラスタに必要なパラメータです。エレメントコンテンツの値を次に示すとおりに指定する必要があります。

   a. tc-properties エレメントに、次のようにプロパティの属性を指定します。

```
<tc-properties>
<property name="ehcache.storageStrategy.dcv2.perElementTTITTL.enabled"
value="true"/>
</tc-properties>
```

`ehcache.storageStrategy.dcv2.perElementTTITTL.enabled` プロパティは、Terracotta サーバが要素ごとの破棄までの時間とアイドルまでの時間の有効期限をサポートするかどうかを決定します。「`true`」値は、クラスタ内の Integration Server で必要で、要素ごとの有効期限を示します。

b. clients エレメントに、次のように logs 子エレメントのコンテンツを指定します。

```
<clients>
<logs>%(com.softwareag.tc.client.logs.directory)</logs>
</clients>
```

このエレメントは、Integration Server がログエントリを書き込むディレクトリを定義します。ログの詳細については、849 ページの「Terracotta Ehcache アクティビティのログへの記録」を参照してください。

3. 変更が終了したら、ファイルを保存して閉じます。

注:
Terracotta Server Array の設定の詳細については、Terracotta の製品マニュアルを参照してください。 *Using BigMemory with webMethods Products* には、サンプルの tc-config.xml ファイルが含まれます。

分散キャッシュの設定に関する考慮事項

分散キャッシュを設定する場合は、以下の点に留意してください。

- 分散キャッシュは、シリアライズ可能なキーと値のみを受け入れます。サービスがシリアライズ可能ではないオブジェクトを実行時にキャッシュに書き込もうとすると、サービスは例外を受け取ります。
- 分散キャッシュは、Integration Server 上ではなく、Terracotta Server Array 上のディスクストアを使用します。Integration Server に存在する分散キャッシュの一部を、ディスクにオーバーフローまたはパーシストするように設定することはできません。Integration Server Administrator を使用して分散キャッシュを作成すると、[**ディスクへオーバーフロー**] または [**ディスクへ保存**] の設定が無効になります。
- Integration Server は、キャッシュの完全修飾名を使用して、Terracotta Server Array のキャッシュを参照します。キャッシュの完全修飾名は、キャッシュマネージャの名前とキャッシュの名前で構成されます。たとえば、「Orders」というキャッシュマネージャの「OrderDetails」というキャッシュの完全修飾名は、「Orders.OrderDetails」です。複数の Integration Server で分散キャッシュを共有している場合は、そのキャッシュに対して同じ完全修飾名を使用する必要があります。
- 複数の Integration Server で共有する分散キャッシュを作成するには、最初に Integration Server のいずれかに分散キャッシュを作成して有効にします。この手順により、Integration Server に分散キャッシュが登録され、Terracotta Server Array にもキャッシュが作成されます。この分散キャッシュを、キャッシュを使用する他の Integration Server に追加します。

これらの Integration Server で分散キャッシュを有効にすると、Terracotta Server Array に既に存在しているキャッシュが認識され、その使用が開始されます。

- 分散キャッシュの特定のパラメータはキャッシュワイドです。つまり、いずれかの Integration Server でパラメータを変更すると、キャッシュを使用するすべての Integration Server に影響します。どのパラメータがキャッシュワイドかの詳細については、818 ページの「分散キャッシュのキャッシュワイドのパラメータとクライアント固有のパラメータ」を参照してください。
- Terracotta Server Array の場合、フェイルオーバーの動作を高可用性 (デフォルト) ではなく一貫性に設定できます。詳細については、『BigMemory Max Administrator Guide』の保証された一貫性へのフェイルオーバーの調整に関するセクションを参照してください。

分散キャッシュのキャッシュワイドのパラメータとクライアント固有のパラメータ

分散キャッシュには、キャッシュワイドのパラメータとクライアント固有のパラメータがあります。

キャッシュワイドのパラメータ

分散キャッシュのキャッシュワイドのパラメータ設定は、Terracotta Server Array で管理され、キャッシュを使用するすべての Integration Server に影響します。1 つの Integration Server でキャッシュワイドのパラメータ設定を変更すると、変更内容はキャッシュを使用するすべての Integration Server に適用されます。

次の表は、キャッシュワイドのパラメータと、これらのパラメータが Terracotta Server Array 上のキャッシュ、Integration Server 上のキャッシュのローカル部分、またはこれら両方のいずれに影響するかを示しています。表に示されているように、キャッシュワイドのパラメータ設定はすべて動的です。つまり、キャッシュワイドのパラメータ設定に対する変更は直ちに有効になります。変更内容を有効にするためにキャッシュマネージャを再初期化したり Terracotta Server Array を再起動したりする必要はありません。

キャッシュワイドのパラメータ	適用対象	動的か
[メモリ内エレメントの最大数]	キャッシュを使用するすべての Integration Server。Terracotta Server Array 自体のキャッシュのサイズには影響しません。	はい
[エターナル]	TSA およびキャッシュを使用するすべての Integration Server 上のキャッシュ。	はい
[廃棄までの時間 (TTL)]	TSA およびキャッシュを使用するすべての Integration Server 上のキャッシュ。 **重要:** キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。	はい

キャッシュワイドのパラメータ	適用対象	動的か
	変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、まず829 ページの「キャッシュマネージャの開始」の手順に従ってキャッシュマネージャを起動してください。	
[アイドルまでの時間]	TSA およびキャッシュを使用するすべての Integration Server 上のキャッシュ。 **重要:** キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、まず829 ページの「キャッシュマネージャの開始」の手順に従ってキャッシュマネージャを起動してください。	はい
[キャッシュ内エントリの最大数]	TSA 上のキャッシュ。Integration Server のキャッシュには影響しません。 **重要:** キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、まず829 ページの「キャッシュマネージャの開始」の手順に従ってキャッシュマネージャを起動してください。	はい

クライアント固有のパラメータ

クライアント固有のパラメータは、特定の Integration Server が分散キャッシュとやりとりする方法に影響します。クライアント固有のパラメータを変更すると、新しい設定は変更を行った Integration Server にのみ影響します。

次の表に示すように、クライアント固有のパラメータには動的パラメータと非動的パラメータがあります。特に指定がない限り、非動的パラメータでは (828 ページの「キャッシュマネージャのシャットダウン」で説明されているように) キャッシュマネージャをシャットダウンする必要があります。変更を加えた後に、(831 ページの「キャッシュマネージャの再初期化」で説明されているように) キャッシュマネージャを再初期化して変更内容を有効にします。以下のパラメータの詳細については、835 ページの「キャッシュの作成」のパラメータの説明を参照してください。

クライアント固有のパラメータ	動的か
[フラッシュ時に消去]	いいえ
[読み取り時にコピー]	いいえ
重要: キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、846 ページの「分散キャッシュの設定の変更」の手順に従います。	
[書き込み時にコピー]	いいえ
[同期書き込み]	いいえ
[一貫性]	いいえ
重要: キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、846 ページの「分散キャッシュの設定の変更」の手順に従います。	
[メモリ保管削除ポリシー]	はい
[高可用性の有効化]	いいえ
[タイムアウト]	はい
[タイムアウトの動作]	はい
[切断時に即時タイムアウト]	はい
[ローカルディスクのエントリの最大数]	はい

注:
技術的には、高可用性に関連する 3 つのパラメータ ([**タイムアウト**]、[**タイムアウトの動作**]、[**切断時に即時タイムアウト**]) は、[**高可用性の有効化**] オプションが有効な場合にのみ、動的に動作します。ただし、Integration Server では [**高可用性の有効化**] は分散キャッシュに対して常に有効であるため、これら 3 つのパラメータは常に動的に動作します。

分散キャッシュの再結合動作

[**再結合**] パラメータを使用すると、切断されたキャッシュマネージャを自動的に Terracotta Server Array と再接続できます。このオプションにより、キャッシュマネージャを手動で再初期化して、切断された Terracotta Server Array との接続を再確立する必要がなくなります。

Integration Server により、[**Terracotta Server Array URL**] パラメータが設定されているすべてのキャッシュマネージャに対して、[**再結合**] パラメータが自動的に有効になります。このキャッシュマネージャの [**再結合**] の設定を無効にすることはできません。

注:
キャッシュマネージャが Terracotta Server Array と再結合すると、新しいクライアントであるかのように配列に結合します。つまり、Integration Server はキャッシュ用に現在ローカルで保持しているデータを消去し、オンヒープストアと、Terracotta Server Array の分散キャッシュからのデータを再同期します。

注:
Integration Server が頻繁に Terracotta Server Array から切断される場合は、OutOfMemory エラーが発生している可能性があります。このエラーは、Integration Server が Terracotta Server Array と再結合するたびに特定のクラス定義が再ロードされるために発生している可能性があります。短期間に多くの再結合が行われる場合は、ガーベッジコレクションが実行される前に、未使用のクラス定義によりヒープの Permanent Generation 領域がいっぱいになっている可能性があります。このエラーの条件、およびそれを回避するために Permanent Generation 領域のサイズを調整する方法の詳細については、Terracotta の製品マニュアルの「Avoiding OOME from Multiple Rejoins」を参照してください。

分散キャッシュの再結合機能の詳細については、BigMemory Max product documentation for 4.1 at www.terracotta.org/documentationの「Rejoin」を参照してください。

Integration Server の [**再結合**] パラメータの詳細については、826 ページの「キャッシュマネージャの作成」を参照してください。

分散キャッシュのノンストップ動作

[**高可用性の有効化**] パラメータは、分散キャッシュをノンストップモードに設定します。ノンストップモードを使用すると、分散キャッシュに対する操作が指定されたタイムアウト時間内に完了しない場合に (たとえば、Terracotta Server Array がビジーで応答しない場合や、ネットワーク接続が切断された場合)、Integration Server は所定のアクションを実行できます。

Integration Server Administrator で分散キャッシュを作成すると、[**高可用性の有効化**] パラメータが自動的に有効になります。このオプションを無効にすることはできません。

Integration Server の分散キャッシュはノンストップモードで動作するため、分散キャッシュを作成するときに以下のパラメータを設定する必要があります。これらのパラメータでは、Integration Server が Terracotta Server Array からの応答を待機する時間や、指定された時間よりキャッシュ操作に時間がかかる場合に実行する手順を指定できます。

パラメータ	説明
[タイムアウト]	Integration Server が Terracotta Server Array に到達できないため、[**タイムアウトの動作**] で指定されたタイムアウト処置を実行する必要があると判断する前に経過する必要がある時間 (ミリ秒単位) を指定します。
[タイムアウトの動作]	Terracotta Server Array が [**タイムアウト**] で指定された時間内に応答しない場合に、キャッシュ操作を完了するために実行する手順を、次のいずれかから指定します。

パラメータ	説明
	■ [**例外**]: NonStopCacheException を返します。 ■ [**NOOP**]: Get 操作に NULL を返します。キャッシュのエレメントを変更する操作 (たとえば、Put 操作、Replace 操作、Remove 操作) を無視します。 ■ [**localReads**]: Get 操作に応答してローカルキャッシュにあるエレメントを返します。キャッシュのエレメントを変更する操作 (たとえば、Put 操作、Replace 操作、Remove 操作) を無視します。
[切断時に即時タイムアウト]	Integration Server が Terracotta Server Array から切断されたことを認識したときに、このキャッシュに対するすべての後続操作が自動的にタイムアウトになるかどうかを指定します。 Integration Server が切断されたときに [**切断時に即時タイムアウト**] オプションが有効になっている場合、Integration Server はキャッシュに対するすべての後続操作を即時にタイムアウトし、[**タイムアウトの動作**] で指定されたタイムアウト処置を実行します。このオプションにより、Integration Server は切断状態にあるときに、すべての操作がタイムアウトするまで待機しなくなります。Integration Server は、Terracotta Server Array と正常に再結合した後、キャッシュ操作の処理を通常どおりに開始します。

分散キャッシュのノンストップ機能の詳細については、BigMemory Max product documentation for 4.1 at www.terracotta.org/documentationを参照してください。

Integration Server でのこれらのパラメータの設定の詳細については、835 ページの「キャッシュの作成」を参照してください。

[キャッシュの最大エントリ数] の値について

Integration Server バージョン 10.0 以降および Integration Server の以前のバージョンの修正で、［キャッシュの最大エントリ数］の値は設定が必須であり、0 より大きい値に設定する必要があります。以前、Integration Server では 0 の値を使用できました。これは、エントリの数がリソースの可用性によってのみ制限されることを示しました。しかし、無制限のキャッシュは無限に大きくなるため、マシンで使用できるリソースが枯渇した場合、パフォーマンスの問題やシステムの障害が発生する可能性があります。

パフォーマンスの問題と OutOfMemoryException を避けるため、Integration Server では、システム分散キャッシュとパブリック分散キャッシュの [キャッシュの最大エントリ数] を 0 より大きい値に設定する必要があります。

- パブリック分散キャッシュの作成または 10.0 より前のバージョンの Integration Server で作成されたパブリック分散キャッシュの編集を行う場合、Integration Server Administrator は [キャッシュの最大エントリ数] の値の妥当性検査を行い、[キャッシュの最大エントリ数] の値が空白であるか 0 (ゼロ) に設定されていると、修正を要求します。10.0 より前のバージョンで作成された既存のパブリック分散キャッシュの場合、Integration Server ではキャッシュが何らかの方法で編集されるまで、[キャッシュの最大エントリ数] を 0 にできます。

- システム分散キャッシュを編集する場合、Integration Server Administrator は [キャッシュの最大エントリ数] の値の妥当性検査を行い、[キャッシュの最大エントリ数] の値が空白であるか 0 (ゼロ) に設定されていると、修正を要求します。システム分散キャッシュの場合、Integration Server ではキャッシュが何らかの方法で編集されるまで、[キャッシュの最大エントリ数] を 0 にできます。
- Integration Server クラスタで使用される既存のシステム分散キャッシュの場合、Integration Server は [キャッシュの最大エントリ数] の値を新しいサーバ設定プロパティ watt.server.cache.maxEntriesInCache で指定された値に設定します。Integration Server の起動時にクラスタリングが有効になっている場合、Integration Server は分散システムキャッシュの値を watt.server.cache.maxEntriesInCache で設定された値に設定します。サーバ設定プロパティが -1 または 0 に設定されている場合、Integration Server は [キャッシュの最大エントリ数] のデフォルト値を設定しません。

キャッシュを検索可能にする

Integration Server は Terracotta Ehcache 検索 API を使用して、キャッシュ内のデータを検索するために使用できるインデックスを構築します。キー、値または値から抽出された属性の事前定義済みのインデックスに対して検索を実行できます。

一部のキーおよび値は、直接検索可能であり、検索インデックスに属性としてそのまま追加できます。ただし、一部のキーまたは値は直接検索できません。直接検索できないキーまたは値は、先に検索可能な属性を抽出する必要があります。

キャッシュを検索可能にしたら、pub.cache:search サービスを使用してキャッシュを検索できます。このサービスは、検索結果をオブジェクト配列として返します。検索結果には、以下のものが含まれます。

- エレメントのキー
- エレメントの値
- エレメントの値から抽出された、事前定義済みの属性値
- 集約の結果。これは集約関数 (Average、Count、Max、Min、Sum など) の結果です。

属性の定義

キャッシュを検索可能にすると、検索で使用する属性を定義できます。属性は、検索時にキーまたは値から抽出されます。キーまたは値から属性を抽出するには、以下の方法を使用できます。

- **式で抽出**キーまたは値から属性を抽出する際の条件となる式を指定します。たとえば、次のように入力します。

  ```
  key.name, value.age
  ```

- **クラスで抽出**エクストラクタクラス、およびオプションでエクストラクタクラスのプロパティを指定します。

  エクストラクタクラスの作成の一般的なガイドラインについては、824 ページの「クラスごとの属性の抽出」を参照してください。サンプルのエクストラクタについては、824 ページの「エクストラクタの例」を参照してください。

キャッシュを検索可能にする場合は、以下の点に留意してください。

- システムキャッシュを検索可能に設定することはできません。
- データが既に含まれているキャッシュを検索可能にすることはできません。

キャッシュを検索可能にする方法の詳細については、835 ページの「キャッシュの作成」を参照してください。

クラスごとの属性の抽出

属性エクストラクタクラスを使用して属性を抽出するには、以下の一般的な手順に従います。

注:
キャッシュでドキュメント (IData) を検索する場合、検索属性を抽出するため、Terracotta Ehcache で使用できる属性エクストラクタクラスを最初に作成する必要があります。Terracotta Ehcache では、pub.cahce:search サービスの実行時に、エクストラクタクラスおよび提供された検索属性情報を使用してキャッシュを検索します。

手順	説明
1	`net.sf.ehcache.search.attribute.AttributeExtractor` の実装である属性エクストラクタクラスを作成します。属性エクストラクタの詳細については、Terracotta Ehcache の製品マニュアルを参照してください。
2	jar ファイルにエクストラクタを入れ、次のいずれか 1 つのディレクトリに jar ファイルを置きます。 ■ *Integration Server_directory* \instances*instance_name* \lib\jars\custom ■ *Integration Server_directory* \instances*instance_name* \packages*packageName* \code\jars\static
3	識別するキャッシュの検索属性を設定します。 ■ クラスによって抽出される属性の名前 ■ エクストラクタクラス ■ エクストラクタクラスのプロパティ (カンマ (,) で区切られたキー/値のペアとして)

エクストラクタの例

次のサンプルは、キー/値のペアで要素の値としてドキュメントを持つ属性を抽出する方法を示します。

```
package com.softwareag.cache.sample ;

import java.util.Properties;
import com.wm.data.IData ;
import com.wm.data.IDataCursor ;
import com.wm.data.IDataUtil  ;
```

```
import net.sf.ehcache.Element;
import net.sf.ehcache.search.attribute.AttributeExtractor;
import net.sf.ehcache.search.attribute.AttributeExtractorException;
/**
*   This attribute extractor is designed for a Doc Type of the following  format
*
*  PO
*    String number
*    Customer
*       String name
*        String address
*
*  In our cache search settings  we  defined  2 search attributes:
*     1) PONumber - Correspo nds to the top level number field
*     2) CustomerName - Corresponds to the customer\name field
*
*/
public  class IDataAttributeExtractor implements AttributeExtractor {
   private Properties _prop = null;

 /**
  *  @param prop Contains the values entered in the "Properties" field for the
 search attribute.
  */
 public IDataAttributeExtractor(Properties prop)
 {
  _prop = prop;
  System.out.println(prop);
 }
 public Object attributeFor(final Element element, final String attributeName)
    throws AttributeExtractorException
 {

  IDataCursor poCursor = null;
  IDataCursor custCursor = null;

  String value = null;
  try
  {
   poCursor  = ((IData)element.getObjectValue()).getCursor();

   if (attributeName.equals("PONumber"))
   {
    // The search attribute PONumber corresponds to the top
    // level number  field in  our  stored document.
    value = IDataUtil.getString (poCursor, "number");
   }
   else if (attributeName.equals("CustomerName"))
   {
    // The search attribute CustomerName corresponds to the
    // customer\name field in our stored document.
    IData custIData = IDataUtil.getIData(poCursor, "customer");
    if(custIData == null)
    {
     //We could not find the customer field in our stored document
     throw new AttributeExtractorException("Unable to find the customer" +
       " field in our element for the " + attributeName + " search attribute.");

    }
```

```
    else
    {
     // return the customer->name value
     custCursor = custIData.getCursor();
     value = IDataUtil.getString(custCursor, attributeName);
    }
   }
   else
   {
    // We do not recognize the search attribute
    throw new AttributeExtractorException("Unknown search attribute: " +
 attributeName);
   }
  }
  finally
  {
   if(poCursor != null) poCursor.destroy();
   if(custCursor != null) custCursor.destroy();
  }

  return value;
 }
}
```

キャッシュマネージャの使用

このセクションでは、キャッシュマネージャを作成、表示、編集、開始、シャットダウン、再ロードおよび削除する方法について説明します。

キャッシュマネージャの作成

キャッシュマネージャを作成するには、以下の手順に従います。

キャッシュマネージャを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. [**キャッシュマネージャの追加**] をクリックします。
4. [キャッシュマネージャの追加] ページの [**名前**] フィールドに、キャッシュマネージャの名前を入力します。

   有効なキャッシュマネージャ名は、以下の条件を満たしている必要があります。

   - NULL 以外であること。
   - `?[ ] / \ = + < > : ; " , * | ^ @` などの文字を使用していないこと。
   - 先頭にピリオド (.)、または末尾にスペースを使用していないこと。

- `SoftwareAG` という文字で始まっていないこと。
- オペレーティングシステムのデバイス名でないこと。

5. このキャッシュマネージャに分散キャッシュが含まれる場合は、[**Terracotta Server Array URL**] フィールドを設定してキャッシュが存在する Terracotta Server Array を指定します。このフィールドには、カンマ区切りのアドレスリスト (*host*:*port* 形式) を、Terracotta Server Array のサーバごとに 1 つ入力します。複数の URL を追加するには、`host1:port,host2:port,...` 形式を使用します。

   注:
   [**Terracotta Server Array URL**] フィールドを指定すると、[**再結合**] チェックボックスが自動的にオンになります。[**再結合**] は、Integration Server が切断時に Terracotta Server Array に自動的に再接続することを示します。このオプションを無効にすることはできません。[**再結合**] オプションの詳細については、820 ページの「分散キャッシュの再結合動作」を参照してください。

6. [**変更内容の保存**] をクリックします。

   キャッシュマネージャを追加する場合は、Integration Server は指定された Terracotta Server Array の URL が有効であることを検証します。Integration Server は、指定された URL が接続を承認することを検証します。URL が有効でない場合は、Integration Server は IOException をスローし、キャッシュマネージャは作成しません。

   注:
   Integration Server は、キャッシュマネージャの作成後、自動的にキャッシュマネージャを開始しません。[**開始**] リンクをクリックして、キャッシュマネージャを開始する必要があります。キャッシュマネージャの開始の詳細については、829 ページの「キャッシュマネージャの開始」を参照してください。

キャッシュマネージャの表示または変更

キャッシュマネージャを表示または変更するには、以下の手順に従います。

注:
この手順の変更手順は、パブリックキャッシュマネージャにのみ適用されます。システムキャッシュマネージャに関連付けられたパラメータを変更することはできません。

キャッシュマネージャを表示または変更するには

1. パブリックキャッシュマネージャの [**Terracotta Server Array URL**] を更新する場合は、続行する前にキャッシュマネージャをシャットダウンするか、または Integration Server を休止モードにします。キャッシュマネージャの実行中はこのパラメータ設定を変更できません。手順については、828 ページの「キャッシュマネージャのシャットダウン」または 983 ページの「保守のためのサーバの休止」を参照してください。

2. Integration Server Administrator を開いていない場合は、それを開きます。

3. [**設定**] > [キャッシュ] に移動します。

4. キャッシュマネージャのリストで、表示または変更するキャッシュマネージャをクリックします。

5. キャッシュマネージャを変更する場合は、[**キャッシュマネージャの設定の編集**] をクリックします (パブリックキャッシュマネージャのみ)。

6. キャッシュマネージャの設定を変更します。

7. [**変更内容の保存**] をクリックします。

8. キャッシュマネージャを再ロードして、変更内容を有効にします。手順の詳細については、829 ページの「キャッシュマネージャの再ロード」を参照してください。

キャッシュマネージャのシャットダウン

キャッシュマネージャが不要になった場合は、シャットダウンしてリソースを解放することができます。

キャッシュマネージャが Terracotta Server Array に接続していない場合は、キャッシュマネージャをシャットダウンするとキャッシュは消去されます。この場合、キャッシュがパーシスタントでないときは、キャッシュのデータは失われます。

キャッシュマネージャが Terracotta Server Array に接続している場合、このキャッシュマネージャに関連付けられている分散キャッシュは、ローカルの Integration Server で実行されているサービスでは使用できません。ただし、これらのキャッシュは Terracotta Server Array に残っているため、他の Integration Server では引き続き使用できます。

注:
この手順は、パブリックキャッシュマネージャにのみ適用されます。システムキャッシュマネージャをシャットダウンすることはできません。

キャッシュマネージャをシャットダウンするには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [キャッシュ] に移動します。

3. [**パブリックキャッシュマネージャ**] で、シャットダウンするキャッシュマネージャの [**シャットダウン**] リンクをクリックします。

4. キャッシュマネージャをシャットダウンするかどうかの確認を求められたら、[**OK**] をクリックします。

キャッシュマネージャの開始

キャッシュマネージャを開始するには、以下の手順に従います。すべてのキャッシュマネージャは、Integration Server の起動時に自動的に開始されます。ただし、Integration Server は、キャッシュマネージャを作成した後には開始しません。以下の手順に従って、新しいキャッシュマネージャを開始する必要があります。また、以下の手順に従って、キャッシュマネージャをシャットダウンした後にキャッシュマネージャを開始することもできます。

注:
この手順は、パブリックキャッシュマネージャにのみ適用されます。システムキャッシュマネージャを手動で開始することはできません。

Integration Server は、キャッシュマネージャを最初に開始するときに、キャッシュマネージャの設定ファイルの設定を使用します。キャッシュマネージャを再実行すると、Integration Server はメモリにキャッシュされた設定情報を使用します。

注:
キャッシュマネージャを開始すると、キャッシュマネージャ内のキャッシュが自動的に有効になります。

キャッシュマネージャを開始するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. [**パブリックキャッシュマネージャ**] リストで、開始するキャッシュマネージャの [**開始**] リンクをクリックします。
4. キャッシュマネージャを開始するかどうかの確認を求められたら、[**OK**] をクリックします。

キャッシュマネージャの再ロード

Integration Server を再起動せずにキャッシュマネージャを再ロードするには、以下の手順に従います。キャッシュマネージャを再ロードする場合は、以下の点に留意してください。

- キャッシュマネージャを再ロードするには、キャッシュマネージャが実行中である必要があります。
- キャッシュマネージャを再ロードすると、ローカルキャッシュが削除されてから再作成されます。キャッシュがパーシスタントでない場合、キャッシュのデータは失われます。
- キャッシュマネージャが Terracotta Server Array に接続している場合、このキャッシュマネージャに関連付けられている分散キャッシュは、再ロードプロセス中にこの Integration Server で実行されているサービスで使用することはできません。ただし、キャッシュマネージャが再ロードされると、キャッシュとそのデータは再び使用できるようになります。

注:

キャッシュマネージャを再ロードすると、キャッシュマネージャ内のキャッシュは自動的に有効になります。

注:
この手順は、パブリックキャッシュマネージャにのみ適用されます。システムキャッシュマネージャを再ロードすることはできません。

キャッシュマネージャを再ロードするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. [パブリックキャッシュマネージャ] で、再ロードするキャッシュマネージャの再ロードアイコン をクリックします。
4. キャッシュマネージャを再ロードするかどうかの確認を求められたら、[**OK**] をクリックします。

キャッシュマネージャの削除

キャッシュマネージャを削除するには、以下の手順に従います。

キャッシュマネージャを削除すると、Integration Server はキャッシュマネージャとそのコンテンツに関連付けられているすべてのキャッシュを削除します。

キャッシュマネージャに分散キャッシュが含まれている場合、これらのキャッシュはこの Integration Server で使用できなくなります。ただし、キャッシュは Terracotta Server Array からは削除されず、Terracotta Server Array に接続されている他の Integration Server で引き続き使用できます。

注:
この手順は、パブリックキャッシュマネージャにのみ適用されます。システムキャッシュマネージャを削除することはできません。

キャッシュマネージャを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. [パブリックキャッシュマネージャ] で、削除するキャッシュマネージャの削除アイコン ✕ をクリックします。
4. キャッシュマネージャを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

キャッシュマネージャの再初期化

キャッシュマネージャまたはそのキャッシュに特定の種類の変更を行う場合、変更内容を有効にするにはキャッシュマネージャを再初期化する必要があります。以下の方法のいずれかでキャッシュマネージャを再初期化できます。

- Integration Server の再起動。
- キャッシュマネージャの開始 (まだ実行されていない場合)。手順の詳細については、829 ページの「キャッシュマネージャの開始」を参照してください。
- キャッシュマネージャの再ロード (既に実行されている場合)。手順の詳細については、829 ページの「キャッシュマネージャの再ロード」を参照してください。

注:
キャッシュマネージャを再ロードする代わりに、キャッシュマネージャを停止して再開できます。この手順は再ロードを実行することと同じです。手順については、828 ページの「キャッシュマネージャのシャットダウン」および 829 ページの「キャッシュマネージャの開始」を参照してください。

Integration Server へのキャッシュマネージャの設定ファイルの追加

キャッシュマネージャの既存の設定ファイルがある場合は (手動で作成したファイルや、別の Integration Server からコピーした設定ファイルなど) 、以下の手順に従って Integration Server に追加することができます。

設定ファイルを別の Integration Server からコピーするのではなく、手動で作成した設定ファイルを追加する場合は、この手順を実行する前に、ファイルが以下の条件を満たしていることを確認します。

- ファイルの名前が拡張子「.xml」で終わっていること。
- ファイルの名前が「SoftwareAG」という文字で始まっていないこと。
- <ehcache> エレメントに name 属性が含まれていること。name 属性では、キャッシュマネージャの名前 (Integration Server に登録されるキャッシュマネージャの名前) を指定します。名前が「SoftwareAG」という文字で始まっていないことを確認します。名前が、ファイルのコピー先である Integration Server 上のキャッシュマネージャで一意であることを確認します。

  例: `<ehcache name="MyCacheManager">`
- キャッシュマネージャが Terracotta Server Array を使用するように設定されている場合は、rejoin 属性が「true」に設定されていること。つまり、<terracottaConfig> エレメントが存在し、その url 属性が指定されている場合は、rejoin 属性も指定され、「true」に設定されている必要があります。

  例: `<terracottaConfig url="srv3A:9510,srv3B:9510" rejoin="true">`

- 設定ファイルに定義されているすべての分散キャッシュに、<nonstop> エレメントが含まれていること。Integration Server では、すべての分散キャッシュがノンストップモードで実行されている必要があります。

  例:

```
<cache
name="JLSCache01"
eternal="true"
maxElementsInMemory="5000"
overflowToDisk="false"
copyOnWrite="true"
copyOnRead="true">
<terracotta> <nonstop></nonstop> </terracotta>
</cache>
```

注:
独自の設定ファイルが既に作成されている場合、Integration Server Administrator では設定ファイルが更新されたときにコメントが保持されないことに注意してください。設定ファイルにコメントが含まれる場合、Integration Server Administrator を使用して最初に編集したときに、これらのコメントは破棄されます。

Integration Server にキャッシュマネージャを手動で追加するには

1. 設定ファイルを *Integration Server_directory*\instances*instance_name*\config\Caching ディレクトリにコピーします。

2. Integration Server を再起動してキャッシュマネージャを初期化します。

   注:
   手順 1 で Integration Server にコピーしたファイルにより、Integration Server に既に登録されているキャッシュマネージャの設定ファイルが置き換えられた場合は、Integration Server を再起動するのではなく、Integration Server Administrator を使用してキャッシュマネージャを再ロードできます(キャッシュマネージャが現在シャットダウンされている場合は、開始してから再ロードします)。手順の詳細については、829 ページの「キャッシュマネージャの再ロード」を参照してください。

3. Integration Server Administrator を開き、**[設定] > [キャッシュ]** に移動して、Integration Server でキャッシュマネージャが正常に追加されたことを確認します。

   キャッシュマネージャが **[設定] > [キャッシュ]** ページに表示されていないか、または正しく機能しない場合は、サーバログまたは Terracotta Ehcache ログを参照して、初期化プロセス中にエラーが発生していないかどうかを確認してください。Terracotta Ehcache ログの表示の詳細については、849 ページの「Terracotta Ehcache アクティビティのログへの記録」を参照してください。設定ファイルに問題がある場合 (Integration Server でファイルを解析できない場合や、ファイルに重複するキャッシュマネージャ名が含まれる場合など) は、Integration Server はサーバログにエラーメッセージを記録し、エラーログに完全なスタックトレースを書き込みます。

キャッシュマネージャの設定ファイルの手動編集

Terracotta Ehcache の設定ファイルの構造に精通している場合は、以下の手順に従って、キャッシュマネージャの設定ファイル内のパラメータを変更できます。パラメータを変更することにより、たとえば、Integration Server Administrator に表示されない特定のキャッシュパラメータを追加または編集できます。

注:
一般的に、キャッシュマネージャまたはキャッシュを変更するには、Integration Server Administrator を使用する必要があります。Integration Server Administrator のユーザインタフェースに表示されないパラメータを編集する必要がある場合にのみ、設定ファイルを手動で編集します。

Integration Server Administrator で生成された設定ファイルにはデフォルト設定を使用するパラメータのエントリが含まれませんので、このファイルを表示するときは注意してください。Integration Server Administrator は、設定ファイルに必須のパラメータ、およびデフォルト値と異なる値のパラメータのみを書き込みます。たとえば、Integration Server Administrator を使用して新しいキャッシュを作成し、[**ディスク有効期限のスレッド間隔**] をデフォルト設定のままにした場合、設定ファイルに <DiskExpiryThreadIntervalSeconds> エレメントは含まれません。[**ディスク有効期限のスレッド間隔**] をデフォルト設定から変更した場合にのみ、設定ファイルにこのエレメントが含められます。

キャッシュマネージャの設定ファイルを手動で編集するには

1. *Integration Server_directory* \instances*instance_name*\config\Caching ディレクトリに移動し、編集する設定ファイルを見つけます。

2. 万一に備えて、続行する前に設定ファイルのバックアップコピーを作成します。

   注:
   Integration Server は、 *Integration Server_directory* \instances*instance_name*\config\Caching ディレクトリで見つけた XML ファイルからキャッシュマネージャを生成しようとするため、バックアップコピーは別のディレクトリに保存するか、「.xml」ファイル拡張子を含まないように名前を変更します。

3. 設定ファイルをテキストエディタで開き、必要に応じて変更します。ファイルを編集するときには、以下の点に留意してください。

   - Integration Server では、Terracotta Server Array に接続しているすべてのキャッシュマネージャで再結合が有効になっている必要があります。rejoin="true" 属性が <terracottaConfig> エレメントに存在する場合は、削除しないでください。
   - Integration Server では、すべての分散キャッシュがノンストップモードで実行されている必要があります。分散キャッシュの定義から <nonstop> オプションは削除しないでください。
   - Integration Server は、キャッシュマネージャの設定ファイル内のコメントを保持しません。編集中に設定ファイルにコメントを追加する場合は、Integration Server Administrator

を使用して設定ファイルを最初に編集するときにコメントが破棄されることに注意してください。

キャッシュマネージャの設定ファイルのパラメータの詳細については、Ehcache product documentation for 2.8 at http://ehcache.org/documentationを参照してください。

4. 設定ファイルを保存した後、設定ファイルを編集したキャッシュマネージャを以下の方法で再初期化します。

   - Integration Server を再起動します。
   - (まだ実行していない場合は) キャッシュマネージャを開始してから、直ちに再ロードします (更新された設定ファイルをディスクから強制的に読み込みます)。手順については、829ページの「キャッシュマネージャの開始」および829ページの「キャッシュマネージャの再ロード」を参照してください。
   - キャッシュマネージャを再ロードします (既に実行されている場合)。手順の詳細については、829 ページの「キャッシュマネージャの再ロード」を参照してください。

5. Integration Server Administrator を使用してキャッシュマネージャとそのキャッシュを表示します。この手順については、827 ページの「キャッシュマネージャの表示または変更」または846 ページの「キャッシュ設定の表示または変更」を参照してください。

   キャッシュマネージャまたはキャッシュが Integration Server Administrator に表示されていないか、または正しく機能しない場合は、サーバログまたは Terracotta Ehcache ログを参照して、初期化プロセス中にエラーが発生していないかどうかを確認してください。Terracotta Ehcache ログの表示の詳細については、849 ページの「Terracotta Ehcache アクティビティのログへの記録」を参照してください。設定ファイルに問題がある場合 (Integration Server でファイルを解析できなかった場合や、ファイルに重複するキャッシュマネージャ名が含まれる場合など) は、Integration Server はサーバログにエラーメッセージを記録し、エラーログに完全なスタックトレースを書き込みます。

6. 分散キャッシュのキャッシュワイドのパラメータのいずれかを変更した場合 ([**メモリ内エレメントの最大数**]、[**エターナル**]、[**廃棄までの時間 (TTL)**]、[**アイドルまでの時間**]、[**キャッシュ内エントリの最大数**]、および [**ログ**] の各パラメータ) は、以下の手順を実行して変更内容を有効にします。

   a. Integration Server Administrator で、[**設定**] > [キャッシュ] に移動します。

   b. 変更したキャッシュが含まれるキャッシュマネージャをクリックします。

   c. [**キャッシュリスト**] で、変更したキャッシュをクリックします。

   d. [**変更内容の保存**] をクリックします。

      この手順により、キャッシュワイドの設定が、この Integration Server および分散キャッシュを使用する他のすべての Integration Server で有効になります。

キャッシュの使用

このセクションでは、キャッシュを作成、編集、消去、無効化、有効化および削除する方法について説明します。

キャッシュの作成

キャッシュを作成するには、以下の手順に従います。

注:
キャッシュをシステムキャッシュマネージャに追加することはできません。

キャッシュを作成する前に、以下の作業を完了する必要があります。

- キャッシュを追加するキャッシュマネージャが既に存在している必要があります。キャッシュマネージャを作成する必要がある場合は、826 ページの「キャッシュマネージャの作成」を参照してください。
- 分散キャッシュを作成する場合は、Integration Server に Terracotta Server Array を使用するためのライセンスが取得されている必要があります。また、キャッシュを追加するキャッシュマネージャが、Terracotta Server Array を使用するように設定されている必要もあります。ライセンスの詳細については、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。
- オフヒープキャッシュ (BigMemory を使用するキャッシュ) を作成する場合は、Integration Server に BigMemory を使用するためのライセンスが取得されている必要があります。ライセンスの詳細については、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。

最初にキャッシュを作成する場合は、開始する前に以下の各セクションを参照してください。

作成するキャッシュ	参照するトピック
オンヒープキャッシュ	809 ページの「オンヒープキャッシュの設定」.
オフヒープキャッシュ	811 ページの「BigMemory キャッシュの設定」.
分散キャッシュ	814 ページの「分散キャッシュの設定」.

注:
以下の手順のパラメータの説明では、パラメータが動的か非動的かを示しています。パラメータが分散キャッシュに適用される場合は、パラメータがキャッシュワイドかクライアント固有かも示しています。 動的パラメータと非動的パラメータの詳細については、806 ページの「動的および非動的キャッシュパラメータ」を参照してください。キャッシュワイドパラメータとクライアント固有パラメータの詳細については、818 ページの「分散キャッシュのキャッシュワイドのパラメータとクライアント固有のパラメータ」を参照してください。

キャッシュを作成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. [**パブリックキャッシュマネージャ**] で、キャッシュを追加するキャッシュマネージャをクリックします。
4. [**キャッシュの追加**] をクリックします。
5. [**キャッシュ設定**] で、以下の情報を指定します。

フィールド	説明
[キャッシュ名]	キャッシュの名前。有効なキャッシュ名は、以下の条件を満たしている必要があります。 ■ キャッシュマネージャ内で一意であること。 ■ NULL 以外であること。 ■ 以下の文字を使用していないこと。 ?[ ] / \ = + < > : ; “ , * | ^ @ **注:** キャッシュを作成した後は、キャッシュの名前を変更することはできません。
[メモリ内エレメントの最大数]	このキャッシュがオンヒープメモリに保持するエレメントの合計数。値 0 は制限なしを示します。ただし、制限なしを指定すると、パフォーマンスに影響する可能性があります。したがって、このフィールドに 0 より大きい値を設定することをお勧めします。 **注:** オフヒープメモリ (BigMemory) を使用している場合は、パフォーマンスの低下を最小限に抑えるために、このフィールドに最低 100 エレメントを設定することをお勧めします。 このパラメータの性質: 動的。キャッシュワイド。
[エターナル]	オプション。オンにすると、このキャッシュのエレメントは、キャッシュに書き込まれた後も期限切れになりません。このオプションを選択した場合は、[**廃棄までの時間 (TTL)**] および [**アイドルまでの時間**] は無視されます。 このチェックボックスは、デフォルトでオフになっています。

フィールド	説明
	このパラメータの性質: 動的。キャッシュワイド。
[廃棄までの時間 (TTL)]	オプション。エレメントがアクセスされているかどうかに関わらず、エレメントをキャッシュに保持できる最大時間 (秒単位)。値 0 は、このキャッシュのエレメントに廃棄までの時間の有効期限がないことを示します。デフォルト値は 0 です。 [**エターナル**] が選択された場合、このフィールドは無視されます。 **注:** [**廃棄までの時間 (TTL)**] および [**アイドルまでの時間**] の両方を 0 に設定した場合、キャッシュのエレメントは期限切れになりません。これらの両方のフィールドを 0 に設定することは、[**エターナル**] チェックボックスをオンにすることと同じです。 **重要:** 分散キャッシュの一部としてキャッシュを作成後にこのフィールドを変更する必要がある場合、キャッシュマネージャがシャットダウンしているときにこの値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、まず829 ページの「キャッシュマネージャの開始」の手順に従ってキャッシュマネージャを起動してください。 このパラメータの性質: 動的。キャッシュワイド。
[アイドルまでの時間]	オプション。アクセスされないエレメントをキャッシュに保持できる最大時間 (秒単位)。値 0 は、このキャッシュのエレメントにアイドルまでの時間の有効期限がないことを示します。デフォルト値は 0 です。 [**エターナル**] が選択された場合、このフィールドは無視されます。 **注:** [**廃棄までの時間 (TTL)**] および [**アイドルまでの時間**] の両方を 0 に設定した場合、キャッシュのエレメントは期限切れになりません。これらの両方のフィールドを 0 に設定することは、[**エターナル**] チェックボックスをオンにすることと同じです。 **重要:** 分散キャッシュの一部としてキャッシュを作成後にこのフィールドを変更する必要がある場合、キャッシュマネージャがシャットダウンしているときにこの値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、まず829 ページの「キャッシュマネージャの開始」の手順に従ってキャッシュマネージャを起動してください。

フィールド	説明
	このパラメータの性質: 動的。キャッシュワイド。
[ディスクへオーバーフロー]	オプション。オンにすると、オンヒープキャッシュおよびオフヒープキャッシュのメモリベースの部分がいっぱいになったときに、ディスクにエレメントが書き込まれます。 このチェックボックスをオンにする場合は、キャッシュのエレメントに使用されるキーと値のペアが Java でシリアライズ可能であることを確認してください。シリアライズ可能でない場合は、パブリックの Integration Server サービスの実行中にキャッシュを使用するときに例外が発生する可能性があります。 このチェックボックスはローカルキャッシュにのみ適用され、デフォルトでオフになっています。[**ディスクへ保存**] チェックボックスがオンになっている場合、分散キャッシュに対しては無効になっています。 このパラメータの性質: 非動的。分散キャッシュには不適用。
[ディスクへ保存]	オプション。オンにすると、Integration Server がシャットダウンされたとき、またはキャッシュマネージャが再初期化されたときに、キャッシュの状態が維持されるようにキャッシュ全体のコピーがディスクに保持されます。 このチェックボックスをオンにする場合は、キャッシュのエレメントに使用されるキーと値のペアが Java でシリアライズ可能であることを確認してください。シリアライズ可能でない場合は、パブリックの Integration Server サービスの実行中にキャッシュを使用するときに例外が発生する可能性があります。 **注:** [**ディスクへ保存**] が有効化されているキャッシュは、キャッシュマネージャが起動されると、別の保存形式には変更できなくなります。 このチェックボックスはローカルキャッシュにのみ適用され、デフォルトでオフになっています。[**ディスクへオーバーフロー**] チェックボックスがオンになっている場合、分散キャッシュに対しては無効になっています。 このパラメータの性質: 非動的。分散キャッシュには不適用。
[ローカルディスクのエントリの最大数]	オプション。このキャッシュ (オンヒープ、オフヒープ、およびディスクのすべて) で保持できるエレメントの最大数。たとえば、5000 エレメントのオンヒープキャッシュがあり、最大 1000 の追加エレメントをディスクにオーバーフローする場合は、[**ローカルディスクのエントリの最大数**] を 6000 に設定します。

フィールド	説明
	値 0 は制限なしを示します。ただし、制限なしを指定するとパフォーマンスに影響する可能性があります。また、ディスクスペースが不足した場合にエラーが発生する可能性もあります。したがって、このフィールドに 0 より大きい値を設定することをお勧めします。 **注:** このフィールドは、ローカルキャッシュにのみ適用され、[**ディスクへオーバーフロー**] チェックボックスがオンになっている場合にのみ有効です。 このパラメータの性質: 動的。クライアント固有。

6. 分散キャッシュを作成する場合は、[**分散**] チェックボックスをオンにして、[**分散キャッシュの設定**] で以下のフィールドを設定します。

注:
[**分散**] チェックボックスをオンにすることができるのは、1) 分散キャッシュの作成に必要なライセンスが Integration Server にあり、さらに 2) キャッシュを追加するキャッシュマネージャで [**Terracotta Server Array URL**] フィールドが設定されている場合のみです。ライセンスの詳細については、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。

フィールド	説明
[一貫性]	クラスタ内のすべてのキャッシュ間でエレメントの一貫性を維持するために使用するモデル。次のいずれかを選択します。 ■ [**強い**] では、クラスタ全体でエレメントに対する更新を直ちに同期することにより、クラスタ間でのエレメントの一貫性が常に保証されます。これがデフォルトです。 ■ [**終了**] では、クラスタ全体でエレメントの最終的な一貫性が保証されます。この設定では、エレメントに対する更新はクラスタ全体ですぐには同期されません。 **重要:** キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、846 ページの「分散キャッシュの設定の変更」の手順に従います。 このパラメータの性質: 非動的。クライアント固有。
[キャッシュ内エントリの最大数]	Terracotta Server Array におけるキャッシュの最大サイズ (エレメントの数)。0 より大きい値を指定する必要があります。

フィールド	説明
	重要: キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、846 ページの「分散キャッシュの設定の変更」の手順に従います。 このパラメータの性質: 動的。キャッシュワイド。
[高可用性の有効化]	オンにすると、分散キャッシュのノンストップ動作が有効になります。ノンストップ動作を使用すると、分散キャッシュに対する操作が指定されたタイムアウト時間内に完了しない場合に (たとえば、Terracotta Server Array がビジーで応答しない場合や、ネットワーク接続が切断された場合)、Integration Server は所定のアクションを実行できます。ノンストップ動作の詳細については、821 ページの「分散キャッシュのノンストップ動作」を参照してください。 **注:** Integration Server では、分散キャッシュの場合、このチェックボックスが自動的にオンになります。このチェックボックスをオフにすることはできません。
[タイムアウト]	オプション。Integration Server が Terracotta Server Array に到達できないため、[**タイムアウトの動作**] で指定されたタイムアウト処置を実行する必要があると判断する前に経過する必要がある時間 (ミリ秒単位)。デフォルト値は 30000 です。 このパラメータの性質: 動的。クライアント固有。
[タイムアウトの動作]	Terracotta Server Array が [**タイムアウト**] で指定された時間内に応答しない場合に、キャッシュ操作を完了するために実行する手順。次のいずれかを選択します。 ■ [**例外**]: NonStopCacheException を返します。これがデフォルトです。 ■ [**NOOP**]: Get 操作に NULL を返します。キャッシュのエレメントを変更する操作 (たとえば、Put 操作、Replace 操作、Remove 操作) を無視します。 ■ [**localReads**]: Get 操作に応答してローカルキャッシュにあるエレメントを返します。キャッシュのエレメントを変更する操作 (たとえば、Put 操作、Replace 操作、Remove 操作) を無視します。 このパラメータの性質: 動的。クライアント固有。

フィールド	説明
[切断時に即時タイムアウト]	オプション。オンにすると、Integration Server が Terracotta Server Array から切断されたときに、このキャッシュに対するすべての後続操作が自動的にタイムアウトになります。 このチェックボックスをオンにすると、Integration Server が切断された場合に、[**タイムアウト**] で指定したタイムアウト時間だけ待機するのではなく、キャッシュに対するすべての後続操作が直ちにタイムアウトになります。Integration Server は、Terracotta Server Array と正常に再結合すると、キャッシュ操作の処理を通常どおり開始します。 このチェックボックスは、デフォルトでオフになっています。 このパラメータの性質: 動的。クライアント固有。
[同期書き込み]	オプション。オンにすると、Terracotta Server Array への同期書き込みが許可されます。このチェックボックスをオンにすると、書き込み操作が完了したと見なす前に、Terracotta Server Array のサーバからの「トランザクション受信済み」確認応答を待機するように Integration Server に要求することにより、データの完全性を最大限に高めることができます。ただし、同期書き込みを有効にすると、分散キャッシュのパフォーマンスが大幅に低下する可能性があります。反対に、このチェックボックスをオフにするとパフォーマンスは向上しますが、データの完全性が低下する可能性があります。 このチェックボックスは、[**一貫性**] が [強い] に設定されている場合にのみ使用できます。デフォルトではオフになっています。 このパラメータの性質: 非動的。クライアント固有。

7. ローカルキャッシュを作成しており、キャッシュを BigMemory にオーバーフローさせる場合は、[**オフヒープへオーバーフロー**] チェックボックスをオンにして、**BigMemory** で以下のフィールドを設定します。

注:
[**オフヒープへオーバーフロー**] チェックボックスは、Integration Server のライセンスで BigMemory の使用が許可されている場合にのみ選択できます。ライセンスの詳細については、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。BigMemory キャッシュの作成に関する考慮事項の詳細については、811 ページの「BigMemory キャッシュの設定」を参照してください。

フィールド	説明
[最大オフヒープ]	このキャッシュで使用可能なオフヒープメモリの最大容量。次のいずれかのサフィックスを容量に追加して、単位を示します。 ■ キロバイトの場合は k または K

フィールド	説明
	■ メガバイトの場合は m または M ■ ギガバイトの場合は g または G ■ テラバイトの場合は t または T たとえば、2g はオフヒープメモリに 2 ギガバイトを割り当てます。割り当て可能な最小容量は 1 メガバイトです。最大容量はありません。 このフィールドは、ローカルキャッシュにのみ適用され、[**オフヒープへオーバーフロー**] がオンになっている場合にのみ使用できます。 このパラメータの性質: 非動的。分散キャッシュには不適用。

8. [**キャッシュの高度な設定**] で以下のフィールドを設定します。

フィールド	説明
[**ディスク有効期限のスレッド間隔**]	オプション。[**廃棄までの時間 (TTL)**] および [**アイドルまでの時間**] に設定された値に基づいて、期限切れエレメントのチェックや削除を行う前に Integration Server が待機する時間 (秒単位)。デフォルト値は 120 秒です。 このフィールドは、ローカルキャッシュにのみ適用され、[**ディスクへオーバーフロー**] チェックボックスがオンになっている場合にのみ有効です。分散キャッシュに対しては無効になっています。 このパラメータの性質: 非動的。分散キャッシュには不適用。
[**ディスクスプールバッファサイズ**]	オプション。このキャッシュが書き込み操作のパフォーマンスを向上させるために使用する、スプールバッファ用にディスクストアに割り当てられたサイズ (メガバイト単位)。デフォルトサイズは 30 メガバイトです。 ディスクストアのパフォーマンスを向上させるにはこのフィールドの値を増やし、OutOfMemory エラーが表示されるようになったら値を減らすことをお勧めします。 このフィールドは、ローカルキャッシュにのみ適用され、[**ディスクへオーバーフロー**] チェックボックスがオンになっている場合にのみ有効です。分散キャッシュに対しては無効になっています。 このパラメータの性質: 非動的。分散キャッシュには不適用。

フィールド	説明
[フラッシュ時に消去]	オプション。オンにすると、キャッシュで Flush 操作が実行された場合にメモリが消去されます。 このチェックボックスは、デフォルトでオンになっています。 このパラメータの性質: 非動的。クライアント固有。
[読み取り時にコピー]	オプション。オンにすると、キャッシュからエレメントが読み込まれるときに、エレメントのコピーが返されます。エレメントに加えた変更は、コピーにのみ影響します。エレメントのコピーをキャッシュに戻すと、変更内容がキャッシュの他のユーザにも見えるようになります。 このチェックボックスをオフにすると、エレメントはコピーされません。エレメントに加えた変更は、キャッシュ内のエレメントを置き換えるために Put 操作が実行されない場合でも、参照を介してエレメントに直接影響します。 このチェックボックスは、デフォルトでオンになっています。 このパラメータの性質: 非動的。クライアント固有。 対応するプロパティは watt.server.serviceResults.copyOnRead です。 **注:** このチェックボックスをオンにすると、アプリケーションでの特定の論理エラーに対してある程度の安全性が確保されます。ただし、キャッシュに対する各 Get 操作によりエレメントのコピーが作成されます。これにより、特にエレメントのサイズが大きい場合は、パフォーマンスに影響することがあります。このため、キャッシュの Get 操作が隔離され、キャッシュされたオブジェクトへの参照がアプリケーションによって保持されないように、アプリケーションコードを構造化することを検討してください。 **重要:** キャッシュマネージャがシャットダウンされている場合は、この値を変更しないでください。変更すると、Integration Server が例外を発行してキャッシュマネージャが起動しなくなる可能性があります。このパラメータを変更するには、846 ページの「分散キャッシュの設定の変更」の手順に従います。
[書き込み時にコピー]	オプション。オンにすると、Put 操作中にキャッシュにエレメントのコピーが配置されます。エレメントに加えた後続の変更は、コピーにのみ影響します。エレメントのコピーをキャッシュに戻すと、変更内容がキャッシュの他のユーザにも見えるようになります。

フィールド	説明
	このチェックボックスをオフにすると、Put 操作では参照を介してエレメントが渡されます。エレメントがキャッシュに書き込まれた後もこのエレメントへの参照を保持すると、その後 Put 操作を実行しない場合でも、エレメントに変更を加えるとキャッシュ内のエレメントも変更されます。 このチェックボックスは、デフォルトでオンになっています。 このパラメータの性質: 非動的。クライアント固有。 対応するプロパティは watt.server.serviceResults.copyOnWrite です。 **注:** このチェックボックスをオンにすると、アプリケーションでの特定の論理エラーに対してある程度の安全性が確保されます。ただし、キャッシュに対する各 Put 操作によりエレメントのコピーが作成されます。これにより、特にエレメントのサイズが大きい場合は、パフォーマンスに影響することがあります。このため、キャッシュの Put 操作が隔離され、キャッシュされたオブジェクトへの参照がアプリケーションによって保持されないように、アプリケーションコードを構造化することを検討してください。
[メモリ保管削除ポリシー]	オプション。[**メモリ内エレメントの最大数**] の値に到達した場合にメモリストアからエレメントを削除するために使用されるポリシー。次のいずれかを選択します。 ■ [LRU (最近使用されていないもの)]。これがデフォルトです。 ■ [LFU (使用頻度が最も低いもの)]。 ■ [FIFO (先入れ先出し)]。この設定は、分散キャッシュには使用できません。 このパラメータの性質: 動的。クライアント固有。
ログ	オプション。オンにすると、基本的な分散マップログメッセージが Terracotta ログに保存されます。 このチェックボックスは、デフォルトでオフになっています。 **注:** 分散キャッシュに対してこのチェックボックスをオンにすると、ログイベントは Terracotta Server Array に記録されます。ログの詳細については、849 ページの「Terracotta Ehcache アクティビティのログへの記録」を参照してください。 このパラメータの性質: 非動的。クライアント固有。

9. キャッシュを検索可能にするには、[**検索可能**] チェックボックスを選択し、[**検索設定**] の以下のフィールドを設定します。

フィールド	説明
[自動インデックスの許可]	オプション。[**キー**]、[**値**]、またはその両方を選択すると、Integration Server によって、検索可能なキーおよび値のインデックスが自動的に作成されます。キーおよび値をクエリーに含めない場合は、自動インデックス作成を有効にしないでください。 **注:** 自動インデックス作成を選択した場合は、キーまたは値の検索タイプを混在させることはできません。 このパラメータの性質: 非動的。キャッシュワイド。

10. [**検索属性**] で、以下のフィールドを定義して、検索で使用される 1 つ以上の属性を定義します。

フィールド	説明
属性	属性の名前。
[抽出メソッド]	キーまたは値から検索属性を抽出するために使用する方法。次のいずれかを選択します。 ■ [**式**] キーまたは値から属性を抽出する際の条件となる式を指定する場合。[**クラス/式**] フィールドに式を入力します。たとえば、「`key.name, value.age`」とします。 ■ [**クラス**] エクストラクタクラス、およびオプションでエクストラクタクラスのプロパティを提供する場合。 [**クラス/式**] フィールドにエクストラクタクラスを入力します。 [**プロパティ**] フィールドに、プロパティをカンマ (,) で区切られたキー/値のペアとして指定します。 エクストラクタは、`net.sf.ehcache.search.attribute.AttributeExtractor` の実装にする必要があります。属性エクストラクタの詳細については、823 ページの「属性の定義」および Terracotta Ehcache の製品マニュアルを参照してください。
[追加/削除]	検索で使用する検索属性を追加するには ＋ を、削除するには ✕ をクリックします。

11. [**変更内容の保存**] をクリックします。Integration Server でキャッシュが作成され、キャッシュは無効な状態になります。

12. キャッシュを実際に使用する場合は、848 ページの「キャッシュの有効化」の手順に従って有効にします。

キャッシュ設定の表示または変更

既存のキャッシュのキャッシュ設定を表示したりキャッシュワイドの設定を変更したりするには、以下の手順に従います。キャッシュマネージャの設定ファイルを手動で編集することで、クライアント固有の設定を変更できます。詳細については、833 ページの「キャッシュマネージャの設定ファイルの手動編集」を参照してください。

注:
この手順では、分散されないキャッシュを変更する方法を説明します。分散キャッシュのキャッシュ設定変更の詳細については、846 ページの「分散キャッシュの設定の変更」を参照してください。

注:
システムキャッシュに関連付けられているパラメータの多くはシステム定義であるため、Integration Server が休止モードで実行されていない場合、Integration Server Administrator では編集できないことに注意してください。システムキャッシュを編集するときは、すべてのパラメータが表示されますが、編集できるのは特定の設定のみです。別の webMethods 製品 (Trading Networks など) のシステムキャッシュを編集している場合は、ガイドラインについて製品資料を参照してください。

キャッシュ設定を表示または変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. 表示または変更するキャッシュが含まれるキャッシュマネージャをクリックします。
4. [**キャッシュリスト**] で、表示または変更するキャッシュをクリックします。
5. 必要に応じて、キャッシュ設定の表示またはキャッシュワイドの設定の変更を行います。

   キャッシュ設定の詳細については、835 ページの「キャッシュの作成」を参照してください。
6. キャッシュ設定に変更を加えた場合は、[**変更の保存**] をクリックして保存します。

分散キャッシュの設定の変更

分散キャッシュの設定を変更するには、以下の手順に従います。

分散キャッシュの設定を変更するには

1. すべてのクライアントでキャッシュマネージャをシャットダウンします。詳細については、828 ページの「キャッシュマネージャのシャットダウン」を参照してください。

2. オフラインデータを削除するには、Terracotta Management Console を使用します。Terracotta Management Console の詳細については、Ehcache product documentation for 2.8 at http://ehcache.org/documentation を参照してください。

3. キャッシュマネージャ設定ファイルで必要な設定を変更します。詳細については、833 ページの「キャッシュマネージャの設定ファイルの手動編集」を参照してください。

4. キャッシュマネージャを再起動します。詳細については、829 ページの「キャッシュマネージャの開始」を参照してください。

キャッシュの無効化

キャッシュを無効にして、たとえば負荷やパフォーマンスの問題のトラブルシューティングを行うことができます。無効なキャッシュに対して実行された Get 操作は NULL を返し、Put 操作は破棄されます。既存のキャッシュエレメントはキャッシュに保持され、キャッシュを有効にしたときに再度使用できます (その間にキャッシュマネージャが再初期化されないことが前提となります)。

キャッシュが分散されている場合は、無効化操作が実行される Integration Server でのみキャッシュが無効になります。キャッシュは引き続き Terracotta Server Array に存在し、Terracotta Server Array に接続されている他の Integration Server からアクセスできます。

注:
無効にすることができるのは、パブリックキャッシュマネージャに属しているキャッシュのみです。

キャッシュを無効化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [キャッシュ] に移動します。

3. [**パブリックキャッシュマネージャ**] で、無効にするキャッシュに関連付けられているキャッシュマネージャをクリックします。

4. [**キャッシュリスト**] で、無効にするキャッシュの横にある [**有効**] 列の [**はい**] リンクをクリックします。

5. キャッシュを無効にするかどうかの確認を求められたら、[**OK**] をクリックします。

キャッシュの有効化

キャッシュが属しているキャッシュマネージャを開始または再ロードすると、キャッシュは自動的に有効になります。Put 操作および Get 操作は、有効になっているキャッシュに対して実行できます。無効になっているキャッシュを有効にすると、キャッシュに以前保存されたすべてのエレメントが再度使用できるようになります (その間にキャッシュマネージャが初期化されていないことが前提となります)。

注:
無効なキャッシュを有効にするには、キャッシュマネージャが実行中である必要があります。キャッシュマネージャがシャットダウンされている間は、キャッシュを有効にすることはできません。

注:
有効にすることができるのは、パブリックキャッシュマネージャに属しているキャッシュのみです。

キャッシュを有効化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。
3. [**パブリックキャッシュマネージャ**] で、有効にするキャッシュに関連付けられているキャッシュマネージャをクリックします。
4. [**キャッシュリスト**] で、有効にするキャッシュの横にある [**有効**] 列の [**いいえ**] リンクをクリックします。
5. キャッシュを有効にするかどうかの確認を求められたら、[**OK**] をクリックします。

キャッシュの消去

キャッシュを消去するときは、そのキャッシュからすべてのエレメントを削除します。

キャッシュが分散されている場合、この手順により Terracotta Server Array 上のそのキャッシュからすべてのエレメントが削除されます。

注:
消去できるのは、パブリックキャッシュマネージャに属しているキャッシュのみです。

キャッシュを消去するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [キャッシュ] に移動します。

3. [**パブリックキャッシュマネージャ**] で、消去するキャッシュに関連付けられているキャッシュマネージャをクリックします。

4. [キャッシュリスト] で、消去するキャッシュの横にある消去アイコンをクリックします。

5. キャッシュを消去するかどうかの確認を求められたら、[**OK**] をクリックします。

注:
消去操作によりエレメントに以前適用されたロックが見つかった場合は、すべてのロックが解放されるまで操作を待機します。このため、ロックが解放されない場合、消去操作は無期限に待機し完了しない可能性があります。

キャッシュの削除

キャッシュを削除すると、キャッシュはキャッシュマネージャから削除されます。キャッシュがローカルの場合、キャッシュ内の既存のエレメントは破棄されます。

キャッシュが分散されている場合、キャッシュが削除された Integration Server ではそのキャッシュが使用できなくなります。このキャッシュは、Terracotta Server Array に接続されている他の Integration Server で引き続き使用できます。

注:
削除できるのは、パブリックキャッシュマネージャに属しているキャッシュのみです。

キャッシュを削除するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [キャッシュ] に移動します。

3. [**パブリックキャッシュマネージャ**] で、削除するキャッシュに関連付けられているキャッシュマネージャをクリックします。

4. [キャッシュリスト] で、削除するキャッシュの横にある削除アイコンをクリックします。

5. キャッシュを削除するかどうかの確認を求められたら、[**OK**] をクリックします。

Terracotta Ehcache アクティビティのログへの記録

Terracotta Ehcache は、SLF4J (Simple Logging Facade for Java) およびその log4J 機能を使用して、Integration Server の Terracotta Ehcache アクティビティに関する情報を記録します。また、キャッシュマネージャが Terracotta Server Array に接続したときに Terracotta Ehcache は追加のログファイルを作成します。

Terracotta Ehcache キャッシュサービスのログへの記録

Terracotta Ehcache は、コンソールにメッセージを記録します。Integration Server は、Terracotta Ehcache ログメッセージを *Integration Server_directory* \instances*instance_name*\logs ディレクトリにある ehcache.log ファイルにリダイレクトします。

Terracotta Ehcache ログにはデフォルトの設定がありますが、tc.dev.log4j.properties ファイルを変更して設定を変更できます。このファイルは *Integration Server_directory* ディレクトリにあります。ログレベルは Warn に設定されています。ログレベルが Warn に設定されていると、Terracotta Ehcache に記録される Fatal、Error、Warn の各メッセージのすべてが含まれます。

注:
log4J の詳細については、http://logging.apache.org/log4j/1.2/index.html に記載されている Apache のログサービスを参照してください。

Terracotta Server Array のキャッシュマネージャアクティビティのログへの記録

Terracotta Ehcache は、トラブルシューティングに役立てるために Software AG Global Support に提供可能なログデータを含む追加のログファイルをサーバに作成します。Terracotta Ehcache は、次の場合にこのログファイルを作成します。

- クラスタ内の Integration Server が起動して、Terracotta Server Array に接続したとき。
- 分散キャッシュが含まれるパブリックキャッシュマネージャが再ロードまたは開始したとき。

Integration Server の watt.server.cachemanager.logsDirectory プロパティを設定して、Terracotta Ehcache がログファイルを保存する場所を指定します。このプロパティのデフォルト値は、*Integration Server_directory* \instances*instance_name*\logs\tc-client-logs です。このプロパティの編集方法の詳細については、146 ページの「拡張設定の使い方」を参照してください。

注:
Integration Server の起動中に、watt.server.cachemanager.logsDirectory プロパティの値が com.softwareag.tc.client.logs.directory プロパティにコピーされます。tc-config.xml ファイルを Terracotta Server Array に設定した場合は、このプロパティを `<client><logs>` 設定に定義しています。tc-config.xml ファイルの設定の詳細については、816 ページの「Terracotta Server Array での tc-config.xml の設定 」を参照してください。

43 拡張 XML パーサの設定

- 拡張 XML パーサとは 852
- 拡張 XML パーサの動作の仕組み 852
- 拡張 XML パーサが使用される状況 854
- 拡張 XML パーサの設定 855
- パーティションサイズの設定 858
- ピーク使用率の統計情報の表示 859

拡張 XML パーサとは

拡張 XML パーサは Integration Server 内の XML パーサで、解析処理中に Terracotta Ehcache を利用します。使用可能なメモリ量に応じて、拡張 XML パーサは Terracotta Ehcache を使用してドキュメントの解析済み部分をローカルディスクストアまたはオフヒープキャッシュ (BigMemory) に移動します。データを移行することで、拡張 XML パーサはヒープ領域を他のドキュメントや他のシステムプロセスに解放します。データをローカルディスクストアまたは BigMemory に移動する機能により、パーサは、複数の大きなサイズの XML ドキュメントなどを、JMV ヒープを独占することなく解析できます。

拡張 XML パーサは DOM ツリーを生成します。具体的には、読み取り専用の org.w3c.dom.Node オブジェクトのツリーを生成します。この DOM ツリーは、XML ドキュメントの特殊な表現方法で、標準の DOM API を使用する任意のプログラムで消費できます。レガシー XML パーサによって生成されたノードを入力として使用していた Integration Server の組み込みサービスは、DOM ノードも同様に使用するように更新されています。Integration Server には、拡張 XML パーサを使用して XML ドキュメントから DOM ノードを作成するためのサービスも含まれています。

それに対して、レガシー XML パーサは、解析処理が終了するまでヒープ内に留まるノードの集合を作成して、XML (または HTML) ドキュメントを解析します。レガシーパーサは com.wm.lang.xml.Node オブジェクトのツリーを生成します。これは、DOM オブジェクトと同様の構造を持つノードのツリーです。

レガシー XML パーサは、解析中にすべてのノードをヒープ上に維持するため、大きなドキュメントまたは複数のドキュメントを同時に解析すると、ヒープ上に多数のオブジェクトが存在することになります。これにより、ガーベッジコレクションの間のパフォーマンスが低下する場合があります。また、レガシー XML パーサが解析中に利用可能なヒープ量を超過した場合、JVM (Java Virtual Machine : Java 仮想マシン) は OutOfMemoryError をスローします。レガシー XML パーサを使用して大きなサイズのドキュメントを処理するために、ヒープサイズを増やしたり、pub.xml 内の queryXMLNode サービスや nodeIterator サービスなどを使用して、ヒープ内に維持されるノードを一部制限したりできます。拡張 XML パーサは、メモリ管理機能を提供することで、その他のサービスを使用することなく、大きなサイズのドキュメント、複数のドキュメントおよび複数の大きなサイズのドキュメントを効率的に処理します。

> 注:
> レガシー XML パーサが XML ドキュメントまたは HTML ドキュメントからノードを作成できるのに対し、拡張 XML パーサは XML ドキュメントだけを解析します。

Terracotta Ehcache および BigMemory の詳細については、799 ページの「Integration Server での Terracotta Ehcache の設定 」を参照してください。

拡張 XML パーサの動作の仕組み

拡張 XML パーサは、JVM ヒープ上にあるパーティションにドキュメントをシリアライズして、そのドキュメントを解析します。ドキュメントの解析に必要なパーティション数およびヒープ領域の量が増加すると、パーサは Terracotta Ehcache を使用してパーティションをローカルディスクストアまたはオフヒープキャッシュ (BigMemory) に移動します。Integration Server は、必要に応じてパーティションを抽出して、後続の処理要求を満たします。

拡張 XML パーサは、ヒープ割り当て制限を使用して、パーティションをローカルディスクストアまたは BigMemory に移行するタイミングを決定します。ヒープ割り当て制限により、拡張 XML パーサはヒープの使用量を調整できます。これにより、Integration Server がピーク容量で動作するときの不安定な動作を抑えることができます。以下の項目を制限できます。

- 1 つの解析対象ドキュメントが使用する合計ヒープ領域。
- すべての解析対象ドキュメントが使用する合計ヒープ領域。
- すべての解析対象ドキュメントが使用する BigMemory の合計量。

以下に、拡張 XML パーサが XML ドキュメントを解析する仕組みの概要を示します。

1. 拡張 XML パーサがドキュメントの解析要求を受信します。ドキュメント解析を開始するために、拡張 XML パーサは同じサイズの 6 個のパーティションをヒープ内に作成します。パーティションのサイズは、サービスに渡される *partitionSize* 値、または拡張 XML パーサに設定されているデフォルトのパーティションサイズで決定されます。パーティションサイズの詳細については、858 ページの「パーティションサイズの設定」を参照してください。

   ただし、6 個のパーティションを作成するためのヒープ領域が確保されていない場合、拡張 XML パーサは十分なヒープ領域が使用可能になるまでドキュメントの解析を遅延します。現在解析されているドキュメントに使用されているヒープ領域の量が [**すべてのドキュメントを組み合わせた場合の最大ヒープ割り当て値**] フィールドで指定されている上限であるか上限に近い場合、十分なヒープ領域が確保できない場合があります。

2. 拡張 XML パーサは、XML ドキュメントをヒープ上のパーティションにシリアライズし、必要に応じて、同じサイズの追加パーティションを作成します。

3. 解析中に、ドキュメントの解析に必要な合計ヒープ領域が、[**任意の単一ドキュメントの最大ヒープ割り当て値**] フィールドで指定された上限または [**すべてのドキュメントを組み合わせた場合の最大ヒープ割り当て値**] フィールドで指定された上限に達した場合、次のいずれかの操作が実行されます。

   - キャッシングが有効化されていない場合、つまりパーサが Terracotta Ehcache を使用してメモリの管理を行わない場合、JVM は OutOfMemoryError をスローします。拡張 XML パーサはドキュメントの解析を完了せず、パーサの呼び出し元サービスは失敗します。
   - キャッシングが有効化されていても BigMemory が有効化されていない場合、拡張 XML パーサは Terracotta Ehcache を使用して一部のパーティションをヒープからローカルディスクストアに移動します。
   - キャッシングが有効化され、BigMemory も有効化されている場合、拡張 XML パーサは Terracotta Ehcache を使用して一部のパーティションをヒープから BigMemory に移動します。拡張 XML パーサが使用する BigMemory の量が [**BigMemory の最大割り当て値**] フィールドで設定された上限を超過した場合、拡張 XML パーサは UnxepectedCachingException をスローします。

非常に稀ですが、拡張 XML パーサが以下の例外をスローする場合があります。

- 任意の 1 つのXML 要素の合計属性値長が 65535 文字を超過した場合、拡張 XML パーサは ImplementationLimitCachingException をスローします。

- パーティションサイズが比較的小さく、個別の XML 要素に非常に多数の属性が含まれているか 1 つの要素の合計属性長が非常に長い場合、Integration Server は PartitionSizeCachingException をスローする場合があります。この問題を解決するには、パーティションサイズを増やします。パーティションサイズの詳細については、858 ページの「パーティションサイズの設定」を参照してください。

拡張 XML パーサが使用される状況

Integration Server は、拡張パーサの使用が明確に指定されている場合のみ、拡張 XML パーサを使用して XML ドキュメントを解析します。拡張 XML パーサは、以下の 3 つの方法のいずれかで、使用することを指定できます。

- pub.xml:loadEnhancedXMLNode を呼び出す。
- pub.xml:xmlStringToEnhancedXMLNode を呼び出す。
- HTTP/S POST 要求を使用する。コンテンツタイプに text/xml または application/xml を指定して要求をターゲットサービスにポストします。さらに、次のいずれかが該当する必要があります。
  - 要求で xmlFormat=enhanced を指定している。
  - 要求では xmlFormat 値が指定されていないが、ターゲットサービスで [**デフォルトの xmlFormat**] プロパティ値が「enhanced」である。
  - 要求では xmlFormat 値が指定されておらず、ターゲットサービス上の [**デフォルトの xmlFormat**] プロパティは空白であり、watt.server.http.xmlFormat が「enhanced」に設定されている。

  HTTP 要求内の xmlFormat および [**デフォルトの xmlFormat**] プロパティの詳細については、『*webMethods Service Development Help*』を参照してください。

レガシー XML パーサを使用していたサービスまたはアプリケーションは、引き続きレガシー XML パーサを使用します。既存のサービスで拡張 XML パーサを使用するには、サービスまたはアプリケーションを変更し、上記のいずれかの方法を使用して拡張 XML パーサを呼び出す必要があります。

注:
既存のサービスまたはアプリケーションについては、Software AG では、`watt.server.http.xmlFormat` パラメータを「enhanced」に変更することによる拡張 XML パーサの使用をお勧めしません。これはグローバルパラメータであり、Integration Server 全体に影響します。サービスまたはアプリケーションベースで xmlFormat を変更するのではなく、すべての Integration Server に対するデフォルトの xmlFormat を変更すると、既存のサービスおよびアプリケーションが失敗する可能性があります。

拡張ノードを消費するサービス

レガシー XML パーサにより生成されるノード (具体的には com.lang.wm.xml.Node タイプのノード) を入力として使用する既存の WmPublic 組み込みサービスの大部分は、拡張 XML パーサにより生成されるノードも使用できます。これには、次のサービスが含まれます。

- pub.xml:xmlNodeToDocument
- pub.xml:queryXMLNode
- pub.xml:getXMLNodeIterator
- pub.xml:freeXMLNode

該当しない例としては、pub.schema:validate サービスがあります。このサービスは、拡張 XML パーサにより生成されたノードの妥当性検査に使用できません。

拡張 XML パーサの設定

拡張 XML パーサを設定するときは、パーサのメモリ使用上限と、使用上限に達した場合にパーティションをローカルディスクストアまたは BigMemory にオーバーフローするかどうかを指定します。

拡張 XML パーサを設定する場合は、以下の点に留意してください。

- 拡張 XML パーサが提供するメモリ管理を利用するには、キャッシングを有効にする必要があります。キャッシングが有効になっている場合、拡張 XML パーサは Terracotta Ehcache を使用して解析に使用されるヒープ量を制限します。キャッシングが有効になっていない場合、拡張 XML パーサはレガシー XML パーサとほぼ同様に動作します。
- キャッシングのみが有効である場合、ヒープ割り当てが上限に達すると、拡張 XML パーサは Terracotta Ehcache を使用してパーティションをローカルディスクストアに移動します。
- キャッシングが有効で、BigMemory も有効である場合、ヒープ割り当てが上限に達すると、拡張 XML パーサは Terracotta Ehcache を使用してパーティションを BigMemory に移動します。
- 拡張 XML パーサで BigMemory を使用するには、キャッシングが有効である必要があります。また、Integration Server に、BigMemory を使用するためのライセンスを取得している必要があります。Integration Server に BigMemory のライセンスがあることを確認し、ライセンスで許可されている BigMemory の量をチェックするには、807 ページの「Terracotta ライセンスのインストール、表示および変更」を参照してください。

拡張 XML パーサを設定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [拡張 XML 解析] に移動します。
3. 拡張 XML パーサがメモリ管理に Terracotta Ehcache を使用する場合は、[**キャッシュ**] の [**有効**] プロパティが [**はい**] に設定されていることを確認します。[**有効**] が [**はい**] に設定されていない場合、[**いいえ**] をクリックして値を変更します。Integration Server Administrator によって、キャッシング機能の有効化の確認を求めるプロンプトが表示されます。[**OK**] をクリックします。

4. 拡張 XML パーサが BigMemory (オフヒープキャッシュ) を利用する場合は、[**BigMemory キャッシュ機能**] の [**有効**] プロパティが [**はい**] に設定されていることを確認します。[**有効**] が [**はい**] に設定されていない場合、[**いいえ**] をクリックして値を変更します。Integration Server Administrator によって、BigMemory のキャッシング機能の有効化の確認を求めるプロンプトが表示されます。[**OK**] をクリックします。

   注:
   拡張 XML パーサで BigMemory を使用するには、キャッシングも有効化する必要があります。

5. [**設定**] > [拡張 XML 解析] > [**拡張 XML 解析設定の編集**] の上部。

6. [**拡張パーサ設定**] の [**デフォルトパーティションサイズ**] フィールドで、拡張 XML パーサが解析済みのドキュメント情報を格納するヒープに対して、パーティションのデフォルトサイズを指定します。次のいずれかのサフィックスを追加して、単位を指定します。

   - キロバイトの場合は k または K
   - メガバイトの場合は m または M

   デフォルトは 20k です。

   注:
   Integration Server は、xmlFormat が「enhanced」に指定された XML ドキュメントをサービスで受信して解析する場合に、デフォルトパーティションサイズを使用します。xmlFormat の詳細については、『*webMethods Service Development Help*』を参照してください。pub.xml:loadEnhancedXMLNode サービスおよび pub.xml:xmlStringToEnhancedXMLNode サービスでは、partitionSize 入力パラメータの値が指定されていない場合は、デフォルトのパーティションサイズを使用します。

7. [**キャッシュ**] で、以下の情報を指定します。

フィールド	指定内容
[すべてのドキュメントを組み合わせた場合の最大ヒープ割り当て値]	ドキュメントを並行して処理するために、拡張 XML パーサが割り当てることのできるオンヒープ領域の最大容量。次のいずれかのサフィックスを容量に追加して、単位を指定します。 ■ キロバイトの場合は k または K ■ メガバイトの場合は m または M ■ ギガバイトの場合は g または G ■ ヒープ領域のパーセンテージを示す場合は % デフォルトでは、ヒープ領域の 40% を JVM で使用できます。 **注:**

フィールド	指定内容
	サフィックスを指定しない場合、Integration Server は単位にバイトを使用します。 解析済みのドキュメントが使用するヒープ全体が [**すべてのドキュメントを組み合わせた場合の最大ヒープ割り当て値**] 値を超えた場合、拡張 XML パーサはヒープ領域が利用可能になるまでドキュメントの解析を開始しません。また、拡張 XML パーサは、一部のパーティションを BigMemory (BigMemory キャッシングが有効な場合) またはローカルディスクストア (BigMemory キャッシングが無効な場合) に移動します。 **重要:** [**すべてのドキュメントを組み合わせた場合の最大ヒープ割り当て値**] をヒープの 100% に設定しないでください。100% の値は、拡張 XML パーサが使用できるヒープの量に制限がないことを示します。多数のドキュメントが同時に解析されたときに、JVM が OutOfMemoryError エラーをスローする場合があります。
[任意の単一ドキュメントの最大ヒープ割り当て値]	1 つのドキュメントを処理するために、拡張 XML パーサが割り当てることのできるオンヒープ領域の最大容量。次のいずれかのサフィックスを容量に追加して、単位を指定します。 ■ キロバイトの場合は k または K ■ メガバイトの場合は m または M ■ ギガバイトの場合は g または G ■ JVM が使用できるヒープ領域の割合を示す場合は % デフォルトはヒープ領域の 20% です。 **注:** サフィックスを指定しない場合、Integration Server は単位にバイトを使用します。 1 つの解析対象ドキュメントが使用するヒープの合計が [**任意の単一ドキュメントの最大ヒープ割り当て値**] 値を超えた場合、拡張 XML パーサは一部のパーティションを BigMemory (BigMemory キャッシングが有効な場合) またはローカルディスクストア (BigMemory キャシングが無効な場合) に移動します。 **重要:** [**任意の単一ドキュメントの最大ヒープ割り当て値**] をヒープの 100% に設定しないでください。100% の値は、拡張 XML パーサが使用できるヒープの量に制限がないことを示します。多数のドキュメントが同時に解析されたときに、JVM が OutOfMemoryError エラーをスローする場合があります。

8. [**BigMemory キャッシュ機能**] の [**BigMemory の最大割り当て値**] フィールドで、拡張 XML パーサでのドキュメントの解析に割り当てることのできる **BigMemory** の最大容量を指定します。

   - キロバイトの場合は k または K
   - メガバイトの場合は m または M
   - ギガバイトの場合は g または G

   デフォルトは 200m です。

   注:
   サフィックスを指定しない場合、Integration Server は単位にバイトを使用します。

   BigMemory 内のパーティションが [**BigMemory の最大割り当て値**] の値で設定された制限を超えた場合、拡張 XML パーサは UnxepectedCachingException をスローします。

9. [**変更内容の保存**] をクリックします。

   変更は、直ちには有効になりません。変更は、拡張 XML パーサにより生成された解析済みドキュメントへの参照がなくなったときに有効になります。

パーティションサイズの設定

拡張 XML パーサがドキュメントを解析するときは、コンテンツを固定サイズのパーティションに配置します。パーサがパーティションのサイズを決定するときに、パーティションサイズを指定して、パーサに判断材料を提供することができます。パーティションサイズの指定は、pub.xml:loadEnhancedXMLNode サービスまたは pub.xml.xmlStringtoEnhancedMLNode サービスの使用時のみ可能です。この両方のサービスには、partitionSize という名前の入力パラメータがあります。パーティションサイズを設定する場合は、以下の点に留意してください。

- partitionSize は、拡張 XML パーサがドキュメントの解析に必要なヒープ領域の量を見積もる際のヒントとなります。多くの場合、解析前に受信 XML ドキュメントのサイズを判断することは不可能です。
- 通例として、Software AG では、partitionSize を未解析 XML ドキュメントのサイズの 1/2 に設定することをお勧めします。
  - 未解析 XML ドキュメントのサイズの 1/2 より大幅に大きい partitionSize を設定すると、拡張 XML パーサは必要な量よりも多くのヒープ領域を消費することになりますが、スループットが向上する場合もあります。ただし、Integration Server の全体的なパフォーマンスに悪影響を与える可能性があります。
  - 未解析 XML ドキュメントのサイズの 1/2 より大幅に小さい partitionSize を設定すると、拡張 XML パーサは多数のパーティションを作成してドキュメントを解析します。これにより、使用するヒープ領域は少なくなりますが、パーサのスループットが低下する場合があります。
  - 未解析 XML ドキュメントのサイズの 1/2 より 3 倍小さい、または 3 倍大きい partitionSize を設定すると、多くの場合、パフォーマンスにほとんど影響しなくなります。

- 実行時に、拡張 XML パーサは、partitionSize を上書きして、使用可能なヒープ領域をすべて消費します。
- 実行時に、partitionSize が [**任意の単一ドキュメントの最大ヒープ割り当て値**] フィールドで設定されている単一ドキュメント制限、または [**すべてのドキュメントを組み合わせた場合の最大ヒープ割り当て値**] で設定されているすべてのドキュメントに対する制限を超える初期ヒープ割り当てとなった場合、拡張 XML パーサはパーティションサイズを自動的に減らします。
- *partitionSize* を指定していない場合、拡張 XML パーサは、Integration Server Administrator の [**設定**] > [拡張 XML 解析] ページの [**デフォルトパーティションサイズ**] フィールドで指定されているデフォルト値を使用します。

拡張 XML パーサが HTTP POST 要求を介して呼び出された場合、拡張 XML パーサは、Integration Server Administrator の [**設定**] > [拡張 XML 解析] ページの [**デフォルトパーティションサイズ**] フィールドで指定されているデフォルトのパーティションサイズを使用します。

ピーク使用率の統計情報の表示

Integration Server では、拡張 XML パーサのピークメモリ使用率の統計情報を追跡できます。この統計情報は、Integration Server の直近の起動時刻から現在までの期間を対象としています。Integration Server は、起動時にこの統計情報をリセットします。

拡張 XML パーサの使用率の統計情報を表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [拡張 XML 解析] に移動します。

   Integration Server Administrator の [ピーク使用統計] に、以下の情報が表示されます。

フィールド	説明
[**すべてのドキュメントに対するヒープ割り当てのピーク**]	拡張 XML パーサが複数のドキュメントの同時解析に使用した最大ヒープ量です。
[**単一ドキュメントに対するヒープ割り当てのピーク**]	拡張 XML パーサが 1 つのドキュメントの解析に使用した最大ヒープ量です。
[**BigMemory 使用のピーク**]	拡張 XML パーサが一度に使用した BigMemory の最大量です。

44 WebSocket の設定

- 概要 862
- WebSocket の動作 862
- WebSocket に対する Integration Server の設定 863
- WebSocket ポートの設定 863
- WebSocketSecure ポートの設定 866
- WebSocket ポート用の WebSocket サーバエンドポイントの表示 869
- [WebSocket サーバセッション] ページ 870
- Integration Server で WebSocket プロトコルを使用する場合の制限 871

概要

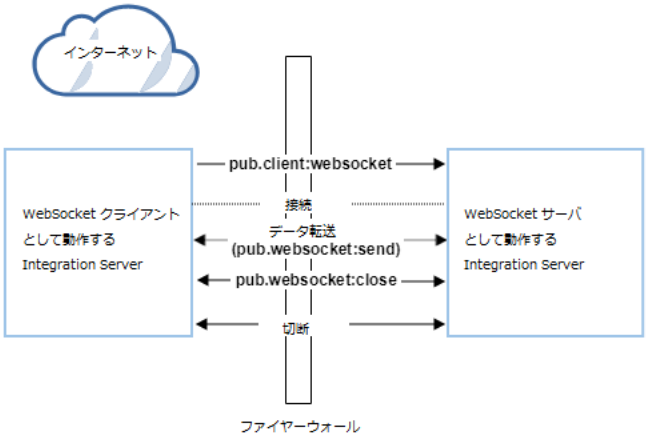
WebSocket プロトコルは、単一の TCP 接続上でクライアントとサーバ/エンドポイント間の同時双方向通信を提供します。Integration Server は WebSocket クライアントエンドポイントと WebSocket サーバエンドポイントとして機能することができます。

WebSocket の動作

WebSocket 設定で、Integration Server は WebSocket サーバまたは WebSocket クライアントとして機能することができます。Integration Server は、WebSocket サーバとして機能する場合、Designer を使用して作成するサーバエンドポイントをホストします。サーバエンドポイントは WebSocket のライフサイクル中にコールバックサービスに必要なプロパティを取得します。エンドポイントはサーバ内のパスによって識別されます。

Integration Server は、WebSocket クライアントとして機能する場合、WebSocket クライアントエンドポイントを作成します。クライアントエンドポイントは、接続およびそれぞれのコールバックサービスに関連する情報を取得します。また、URI によって識別される WebSocket サーバに接続します。

次の図は、要求が WebSocket 設定で処理される仕組みを示したものです。



(1)クライアントと Integration Server 間の WebSocket セッションを確立するには、クライアントが HTTP アップグレード要求を送信する必要があります。それには、`pub.client:websocket` サービスを使用します。

(2)Integration Server では、セッションのアップグレードが可能であり、それが望ましい場合、HTTP 101 応答を要求元クライアントに送信します。この時点でハンドシェイクは成功したと見なされ、サーバとクライアント間の接続は WebSocket プロトコルにアップグレードされます。それには、使用可能な WebSocket ポートを使用します。

(3)WebSocket セッション上で、Integration Server とクライアント間でデータを双方向に流すことができます。それには、`pub.websocket:send` サービスを使用します。

(4)データ交換を行うクライアントまたはサーバは、終了要求を送信することで、WebSocket 接続の終了を要求できます。それには、`pub.websocket:close` サービスを使用します。

WebSocket に対する Integration Server の設定

ここでは、WebSocket アプリケーションを設定するための高レベルの手順を説明します。次のチェックリストは、これらの手順を要約したものです。

☒	タスク	メモ
	WebSocket ポートの設定	必要に応じて、WebSocket または WebSocketSecure ポートオプションを使用できます。手順については、『863 ページの「WebSocket ポートの設定」』を参照してください。 **注:** WebSocketSecure ポートを使用する場合は、サーバ認証、サーバの秘密鍵および認証局の認証をこのサーバに保存する必要があります。手順については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。
	WebSocket サーバエンドポイントの設定 (オプション) Integration Server が Websocket サーバとして機能している場合	このタスクの詳細については、『webMethods Service Development Help』を参照してください。
	WebSocket クライアントエンドポイントの設定 (オプション) Integration Server が Websocket クライアントとして機能している場合	このタスクの詳細については、『webMethods Service Development Help』を参照してください。

WebSocket ポートの設定

WebSocket サーバとして機能する Integration Server は、ポートを使用して、接続を維持するクライアントからの要求を受信待機します。

WebSocket ポートを追加するには、以下の手順に従います。

WebSocket ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [ポート] に移動します。
3. [**ポートの追加**] をクリックします。

4. [**ポートの追加**] ページで、[**webMethods/WebSocket**] を選択します。

5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server Administrator のページが表示されます。

6. [**WebSocket ポート設定の編集**] ページで、[**webMethods/WebSocket**] ポートの下に以下の情報を入力します。

パラメータ	指定内容
[有効]	この WebSocket リスナーを有効にするか ([**はい**]) 無効にするか ([**いいえ**]) を指定します。
[ポート]	ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバで使用されるポート番号がそれぞれ固有であることを確認してください。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[バインドアドレス(オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Integration Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 6000 です。最大値は 65535 です。

パラメータ	指定内容
	注: [バックログ] パラメータは無効化されたポートには適用されません。Integration Server は、無効化されたポートへ送信する要求を拒否します。

7. [**スレッドプール設定**] で、次の情報を入力します。

注:
Integration Server はこのポートで受信したプライベートスレッドプールを使用します。このスレッドオプションを使用すれば、このポートで受信した要求がスレッドの他のサーバ機能と競合することを回避できます。

パラメータ	指定内容
[スレッドプールの最小値]	Integration Server がサーバスレッドプールで保持するスレッドの最小数。デフォルトは 10 です。
[スレッドプールの最大値]	Integration Server がサーバスレッドプールで保持するスレッドの最大数。デフォルトは 200 です。
[スレッドプールの優先順位]	Java がこのスレッドプールのスレッドに割り当てる優先順位。数が大きいほど、優先順位が高くなります。デフォルトは 5 です。 **重要:** この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。 ポートの詳細を表示すると、そのポートで現在使用中のプライベートスレッドプールスレッドの合計数が Integration Server から報告されます。

8. [**WebSocket ポリシー**] で、次の情報を入力します。

パラメータ	指定内容
[アイドルタイムアウト (ミリ秒)]	このタイムアウト値内でサーバがクライアントからの要求を受信しなかった場合に WebSocket クライアント接続を閉じる時間 (300000 ミリ秒)。

9. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定内容
[Client Authentication]	この WebSocket ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。Integration Server はユーザ名とパスワードを使用します。

10. [**変更内容の保存**] をクリックします。

11. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

    ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。519 ページの「リソースへのアクセスをポートごとに制御」

12. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

WebSocketSecure ポートの設定

WebSocketSecure ポートによって、Integration Server は接続のセキュリティを確保できます。デフォルトでは、WebSocket リスナーはデフォルトの Integration Server SSL キーの認証を使用します。ただし、Integration Server キーストア内にあるリスナー自体の秘密鍵を使用するようにリスナーを設定できます (ファイルベースまたは SmartCard/HSM ベース)。詳細については、492 ページの「Integration Server の SSL キーと認証の設定」を参照してください。

WebSocketSecure ポートを追加するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. [**ポートの追加**] をクリックします。

4. [**ポートの追加**] ページで、[**webMethods/WebSocketSecure**] を選択します。

5. [**サブミット**] をクリックします。ポート情報の入力を要求する Integration Server Administrator のページが表示されます。

6. [**WebSocket リスナー設定**] で、次の情報を入力します。

パラメータ	指定内容
[有効]	この WebSocketSecure リスナーを有効にするか ([**はい**]) 無効にするか ([**いいえ**]) を指定します。

パラメータ	指定内容
[ポート]	ポートに使用する番号。このホストマシンでまだ使われていない番号を選択します。 **重要:** 同じホストマシン上で複数の Integration Server を実行している場合、各サーバで使用されるポート番号がそれぞれ固有であることを確認してください。
[エイリアス]	この Integration Server で一意であるポートエイリアス。エイリアスは 1～255 文字の長さで指定し、文字 (a～z、A～Z)、数字 (0～9)、アンダースコア (_)、ピリオド (.)、ハイフン (-) を 1 文字以上使用する必要があります。
[説明]	ポートの説明。
[パッケージ名]	このポートに関連付けるパッケージ。パッケージを有効にすると、ポートも有効になります。パッケージを無効にすると、ポートも無効になります。 このパッケージを複製すると、Integration Server によって、同じ番号、同じ設定のポートがターゲットサーバに作成されます。同じ番号のポートが既にターゲットサーバに存在する場合、ポートの設定は元のままになります。この機能は、特定のポートで入力を必要とするアプリケーションを作成する場合に便利です。この場合、アプリケーションは別のサーバに複製された後も引き続き稼動します。
[バインドアドレス(オプション)]	このポートをバインドする IP アドレス。バインドアドレスは、使用するコンピュータに複数の IP アドレスが設定されており、かつ、ポートで特定のアドレスを使用する場合に指定します。バインドアドレスを指定しない場合は、アドレスが自動的に指定されます。
[バックログ]	Integration Server で要求の拒否が開始される前に、有効化されているポートのキューに保持される要求の数。デフォルトは 6000 です。最大値は 65535 です。 **注:** [バックログ] パラメータは無効化されたポートには適用されません。Integration Server は、無効化されたポートへ送信する要求を拒否します。

7. [**スレッドプール設定**] で、次の情報を入力します。

パラメータ	指定内容
[スレッドプールの最小値]	Integration Server がサーバスレッドプールで保持するスレッドの最小数。デフォルトは 10 です。

パラメータ	指定内容
[スレッドプールの最大値]	Integration Server がサーバスレッドプールで保持するスレッドの最大数。デフォルトは 200 です。
[スレッドプールの優先順位]	Javaがこのスレッドプールのスレッドに割り当てる優先順位。数が大きいほど、優先順位が高くなります。デフォルトは 5 です。 デフォルトは 5 です。 **重要:** この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。

8. [**WebSocket ポリシー**] で、次の情報を入力します。

パラメータ	指定内容
[アイドルタイムアウト(ミリ秒)]	このタイムアウト値内でサーバがクライアントからの要求を受信しなかった場合に WebSocket クライアント接続を閉じる時間 (300000 ミリ秒)。

9. [**セキュリティ設定**] で、次の情報を入力します。

パラメータ	指定内容
[Client Authentication]	この WebSocket ポートに到着した要求に対して Integration Server が実行するクライアント認証のタイプ。Integration Server はユーザ名/パスワードを使用します。

10. [**リスナー固有のクレデンシャル (オプション)**] で、次の情報を入力します。

注:
[認証] ページで指定した認証とは異なる認証セットを使用する場合にのみ、この設定を使用します。

パラメータ	指定内容
[キーストアエイリアス]	オプション。Integration Server キーストアのユーザ指定のテキスト識別子。 エイリアスは秘密鍵および関連する認証のリポジトリを指し示します。それぞれのリスナーで 1 つのキーストアを指し示しますが、同じキーストア内に複数のキーおよび認証が存在し、複数のリスナーで同じキーストアエイリアスを使用することもできます。

パラメータ	指定内容
	詳細については、513ページの「キーストアエイリアスの作成」を参照してください。
[キーエイリアス]	オプション。上記のキーストアエイリアスによって指定されるキーストアに保存されている必要がある、秘密鍵のエイリアス。

11. [**変更内容の保存**] をクリックします。

12. 必要であれば、[**ポート**] ページの [**編集**] をクリックして、アクセスモードを変更します。**アクセスモードのデフォルトを許可に設定**するか、**デフォルトのアクセス設定値にリセット**できます。

    注:
    ポートが WebSocket または WebSocketSecure のものである場合、WebSocket エンドポイントが表示されます。

    ポートのアクセスモードの設定およびポートへの IP アクセスの制御の詳細については、以下を参照してください。 519 ページの「リソースへのアクセスをポートごとに制御」

13. [**ポート**] ページのポートの一覧内にある [**有効**] 列の状態が [**はい**] になっていることも確認します。有効になっていない場合は、[**いいえ**] をクリックしてポートを有効にします。

WebSocket ポート用の WebSocket サーバエンドポイントの表示

WebSocket ポートについて、WebSocket サーバエンドポイントのリストを表示できます。

WebSocket ポートへの接続を表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [ポート] に移動します。

3. WebSocket ポートのポート番号をクリックします。

4. [**WebSocket ポートの詳細の表示**] ページで、[**WebSocket サーバエンドポイント**] をクリックします。

   Integration Server Administrator は [**WebSocket サーバエンドポイント**] ページを表示します。[**WebSocket - 登録済みサーバエンドポイント**] テーブルはそのポートの WebSocket サーバエンドポイントを示します。Integration Server Administrator には、接続ごとに以下の情報が表示されます。

フィールド	説明
[WebSocket URI]	WebSocket クライアントが接続に使用するサーバエンドポイントを示します。
[アクティブセッション]	サーバでアクティブなセッションの数を示します。セッションの現在数をクリックすると、[**WebSocket サーバセッション**] ページが表示されます。

注:
アクティブなセッションがない場合、[**WebSocket ポート <portNumber> にアクティブなセッションは見つかりません。**] が表示されます。

[WebSocket サーバセッション] ページ

[WebSocket サーバセッション] ページを使用して、現在アクティブな WebSocket サーバセッションを表示できます。セッションの現在数をクリックすると、次の情報が表示されます。

フィールド	説明
[セッション ID]	ホストセッション ID の値が表示されます。
[ローカルアドレス]	その特定セッションに対する WebSocket サーバのローカルアドレスとポートが表示されます。
[リモートアドレス]	その特定セッションに対するクライアントのリモートアドレスとポートが表示されます。
[プロトコルバージョン]	WebSocket プロトコルバージョンが表示されます。Integration Server では WebSocket プロトコルバージョン 13 のみサポートしています。
[アイドルタイムアウト]	WebSocket が未使用の接続をオープンしておくことのできる時間 (ミリ秒) を示します。
[セキュア]	WebSocket セッションがセキュアであるかどうかを示します。[**はい**] はセッションがセキュアであることを示します。[**いいえ**] はセッションがセキュアでないことを示します。

WebSocket ポートの WebSocket サーバセッションを管理するには、次のオプションを使用します。

オプション	説明
[閉じる]	選択した WebSocket セッションを閉じます。
[切断]	選択した WebSocket セッションを切断します。

Integration Server で WebSocket プロトコルを使用する場合の制限

Integration Server WebSocket モジュールでは、Websocket の次の機能をサポートしていません。

- JWT (JSON Web Token) による認証
- リバースプロキシとフォワードプロキシ
- 非同期送信のサポート (ブロッキング送信のみサポートされます)
- WebSocket による分割メッセージのサポート
- Ping/Pong フレーム
- 逆呼び出し

45 ロックの管理および最適な実例

- はじめに 874
- ローカルサーバロックまたは VCS 統合ロックの選択 874
- ロックの無効化および再有効化 874
- 最適な実例 876

はじめに

この章で提供する情報は、サーバ管理者および運用プロセスの一環としてパッケージを定期的に複製およびパブリッシュするユーザを対象としています。

ローカルサーバロックまたは VCS 統合ロックの選択

次の 2 つのロック形式のいずれかをサポートするために webMethods Integration Server を設定できます。

- Integration Server のファイルシステム内で適用するローカルロック。
- サードパーティ製の VCS (Version Control System : バージョン管理システム) リポジトリとの統合の結果としてのロック。この場合は、VCS リポジトリに対するチェックアウトおよびチェックインの際にエレメントのロックおよびアンロックが行われます。

この章では、Integration Server のローカルロックによるロック管理タスクについて説明します。サーバ環境でバージョン管理システムを使用していない場合は、この章の説明に従って、Integration Server のローカルロックを設定してください。

Designer のローカルサービス開発機能を使用して、サードパーティ製のバージョン管理システムを使用できます。ローカルサービス開発の詳細については、『*webMethods Service Development Help*』を参照してください。

ロックの無効化および再有効化

Integration Serverにロックを実装しない方がよい場合があります。サーバ管理者の場合、*Integration Server_directory* \instances*instance_name*\config\server.cnf にある設定パラメータを編集することで、ロックを無効化または再有効化できます。

ロックを無効化または再有効化する前に、次の作業を完了する必要があります。

- すべてのユーザがサーバ上での開発作業を完了し、すべてのエレメントがアンロックされていることを確認します。
- Designer のすべてのセッションを閉じます。以下の手順で管理者が拡張設定を変更した後、ユーザは Designer の新規セッションを開く必要があります。

ロックの無効化

ロックを無効にするには、Integration Server Administrator を使用して拡張設定を編集するか、server.cnf を手動で編集します。以下の手順では、Integration Server Administrator を使用する場合の手順について説明します。

重要:
この方法で設定変更を行う場合は、server.cnf の編集による変更は行わないでください。後で server.cnf を編集して設定を変更すると、競合が発生する可能性があります。

Integration Server でロックを無効化するには

1. 874 ページの「ロックの無効化および再有効化」で説明したタスクを完了します。
2. [**設定**] > [拡張設定] に移動します。
3. [**拡張設定の編集**] をクリックします。
4. 次の表に従って、[拡張設定] ボックスにキーと値を入力します。

目的	入力する文字列
ユーザロックを無効化し、ロックを表示しない	watt.server.ns.lockingMode=none
ユーザロックは無効化するが、システムロックは表示する	watt.server.ns.lockingMode=system

5. [**変更内容の保存**] をクリックします。変更した情報は *Integration Server_directory* \instances*instance_name*\config\server.cnf に保存されます。
6. Integration Server を再起動します。更新した設定が有効になります。

ロックの再有効化

ロックを再有効化するには、Integration Server Administrator を使用するか、server.cnf を手動で編集します。以下の手順では、Integration Server Administrator を使用する場合の手順について説明します。

重要:
この方法で設定変更を行う場合は、server.cnf の編集による変更は行わないでください。後で server.cnf を編集して設定を変更すると、競合が発生する可能性があります。

Integration Server でロックを再有効化するには

1. 874 ページの「ロックの無効化および再有効化」で説明したタスクを完了します。
2. [**設定**] > [拡張設定] に移動します。
3. [**拡張設定の編集**] をクリックします。
4. [拡張設定] ボックスで、「watt.server.ns.lockingMode」の値を「**full**」に設定します。
5. [**変更内容の保存**] をクリックします。変更した情報は *Integration Server_directory* \instances*instance_name*\config\server.cnf に保存されます。
6. 変更した内容を有効にするために、Integration Server を再起動します。

最適な実例

リモートサーバの設定

- Integration Server クラスタでは、共同開発機能を使用しないことをお勧めします。この機能を使用すると、クラスタ内の他の Integration Server との間で、エレメントのロック情報を不用意に共有してしまう可能性があります。このような共同開発の問題を防止するために、開発中はクラスタではなく、スタンドアロンの Integration Server を使用してください。

サーバのユーザ名

- Integration Server にログオンするときには、固有のユーザ名を使用してください。ロックはユーザ名に基づいているため、ユーザごとに一意のユーザ名でサーバにログオンすることが重要です (「Administrator」や「Developer」でログオンしないでください)。

パッケージの複製およびパブリッシュ

- パッケージの複製とパブリッシュを使用して定期的に (毎日または毎晩) パッケージをバックアップしてください。パッケージを複製した場合、ロック情報はパッケージ (または部分パッケージ) と共には移動されないため、日付に基づいて各パッケージにバージョン番号を付けておくことをお勧めします。パッケージの置換や上書きはしないでください。パッケージを置き換える必要がある場合は、古いパッケージを完全に削除してから新しいパッケージをインストールしてください。

  注:
  パッケージを置換または上書きすると、webMethods Integration Server によってエレメントの共通部分がナビゲーションパネルに取り込まれます。また、既存のパッケージはサーバインスタンスの replicate\salvage フォルダに移動されます。

- パッケージを複製およびパブリッシュした場合、ロック情報は保持されません。これは予期される動作であり、複製およびパブリッシュ機能の設計の一部でもあります。ただし、システムロックは保持されます (読み取り専用ファイル属性)。

- パッケージをパブリッシュするときには、ユーザロックは保持されないことに注意してください。

- 削除したパッケージを復元しても、ロック情報は保持されません。パッケージを復元または削除するときには、復元先のパッケージですべてのロックが削除されることに注意してください。

  頻繁に複製または部分複製するパッケージの場合、システムロックやユーザロックは使用しないことをお勧めします。たとえば、パッケージを頻繁に更新してパートナに送信する場合がこのケースに該当します。

パッケージとフォルダの編成

- 開発者ごとまたは Java/C サービスごとに、単一パッケージまたは単一フォルダを使用してください。

ソースコード

- ソースコードに重要な変更が加えられた場合は、必ずパッケージを再ロードして、最新のシステムロックを反映させてください。

webMethods Integration Server のアップグレード

- webMethods Integration Server を新しいバージョンにアップグレードすると、すべてのロック情報が失われます。したがって、アップグレードする前に、すべてのロックが削除され、すべての変更が保存されていることを確認してください。

46 webMethods Messaging Trigger の管理

- はじめに 880
- ドキュメント抽出の管理 880
- ドキュメント処理の管理 890
- webMethods Messaging Triggerに使用されるサーバスレッド数の制限 898
- webMethods Messaging Trigger管理のクラスタ同期 900
- webMethods Messaging Triggerプロパティの変更 903
- トリガーサービスの再試行およびシャットダウン要求の管理 904
- webMethods Messaging Triggerのポーリング要求の遅延 906
- Integration Server の以前のバージョンから 10.3 以降への逐次トリガーの移行 909

はじめに

パブリッシュ/サブスクライブソリューションでは、Integration Server によるドキュメントの抽出やフローによってリソース (主にサーバスレッドとメモリ) が消費されます。Integration Server がドキュメントを抽出し処理する速度は、こうしたリソースがどの程度利用可能かによって決まります。ただし、その他の機能の実行にもサーバリソースが必要です。Integration Server Administrator には、ドキュメントの抽出や処理に消費されるリソースを管理するための制御が備わっています。これらの制御を使用することによって、ドキュメントの抽出や処理に必要なリソースと、その他の機能の実行に必要なサーバリソースとのバランスをとることができます。

具体的には、Integration Server Administrator に備わっている制御を使用して次のようなことができます。

- Broker からのドキュメント抽出に使用されるサーバスレッド数を増加または減少させる
- webMethods messaging triggerキューの容量を減らす。
- 1 つまたは複数の webMethods messaging triggerのドキュメント抽出を一時停止する。
- ドキュメント処理に使用されるサーバスレッド数を増加または減少させる。
- Integration Server で同時実行 webMethods messaging triggerのドキュメントを並行処理するために使用できるスレッド数を減少させる。
- 1 つまたは複数の webMethods messaging triggerのドキュメント処理を一時停止する。
- 特定の webMethods messaging triggerのトリガーキューの容量、補充レベルまたは実行スレッド数を変更する。

さらに、Integration Server Administrator にはクラスタ同期機能があり、これを使用することによって、選択した変更内容をクラスタの他の Integration Server に自動的に反映させることができます。

これらの制御により、サーバスレッドやメモリを解放して予想外のサーバ負荷 (HTTP 要求が急に殺到するなど) に対応したり、高トラフィックが予想される時間帯に対応したりすることができるため、実稼動環境で役に立ちます。また、これらの制御をプロジェクトのキャパシティプランニング時に使用して、webMethods messaging triggerおよびサーバスレッドの使用数に関する設定値を決定することもできます。

以下の各項では、これらの制御を使用してドキュメント抽出やドキュメント処理を管理する方法について詳しく説明します。

注:
Integration Server バージョン 9.5 SP1 よりも前の製品では、webMethods messaging trigger は Broker/ローカルトリガーと呼ばれていました。

ドキュメント抽出の管理

パブリッシュ/サブスクライブモデルにおいて、「ドキュメント抽出」とは、Integration Server がサーバスレッドを使用してメッセージングプロバイダから追加のドキュメントをフェッチするプロセスを意味します。ドキュメント抽出には、ドキュメントを要求してメッセージングプロバ

イダから抽出するためのサーバスレッドが必要です。さらに、Integration Server は抽出するドキュメントの一時コピーをメモリに保持するので、ドキュメント抽出にはメモリも必要です。ドキュメントの処理が正常に終了すると、Integration Server は一時コピーをメモリから解放します。

Integration Server には、ドキュメント抽出用のサーバリソースを管理するための制御が備わっています。具体的には、これらの制御を使用して次のようなことができます。

- トリガーキューの容量を調整することによって、ドキュメント抽出の速度や抽出時のメモリ使用量を管理する。
- 1 つまたは複数の webMethods messaging triggerのドキュメント抽出を一時停止または再開する。
- Broker からのドキュメント抽出に使用されるサーバスレッド数を制限する。

これらの制御は、開発時、キャパシティプランニング時または実行時に使用できます。以下の項目では、これらの制御について詳しく説明します。

Integration Server が各トリガークライアントキューをポーリングして、Broker で新しいドキュメントの有無を確認する頻度を設定することもできます。詳細については、906 ページの「webMethods Messaging Triggerのポーリング要求の遅延」を参照してください。

webMethods Brokerからのドキュメント抽出に使用するスレッドの増減

実稼動環境やキャパシティプランニングで、Broker からドキュメントを抽出するのに使用するスレッド数を増加または減少させることができます。デフォルトでは、Integration Server はサーバスレッドプールを最大で 100% 使用して Broker からドキュメントを抽出します。各 webMethods messaging triggerは、別個のサーバスレッドを使用して Broker からドキュメントを抽出します。たとえば、サーバスレッドプールの最大サイズが 80 スレッドで、サーバスレッドプールを 100% 使用してドキュメントを抽出できる場合、一度に最大 80 のトリガーでドキュメントを要求することができます。

ドキュメント抽出に使用できるサーバスレッドのパーセンテージを指定することによって、ドキュメント抽出に使用されるスレッドの最大数を制限できます。Integration Server は、指定されたパーセンテージを使用して Broker からのドキュメント抽出に使用できるサーバスレッド数を計算します。

たとえば、サーバスレッドプールの最大サイズが 80 スレッドであるとします。ドキュメント抽出スレッドの最大値に対するパーセンテージを 10% に指定した場合、Integration Server が Brokerからのドキュメント抽出のために一度に使用できるスレッドの数は 8 つだけになります。Integration Server は webMethods messaging triggerごとに別々のスレッドを使用してドキュメントを抽出するため、Integration Server は、Broker からドキュメントを受信する 8 トリガー分しか一度にドキュメントを抽出できないことになります。

Broker からのドキュメント抽出に使用されるサーバスレッドプールのパーセンテージを減らすと、ドキュメントを同時に抽出できるトリガーが少なくなるため、ドキュメント抽出の速度が遅くなる可能性があります。また、HTTP 要求への応答、Universal Messaging からのドキュメント抽出、ドキュメント処理などの他のタスクでサーバスレッドを使用できるようになります。

Broker からのドキュメント抽出に使用されるサーバスレッドプールのパーセンテージを増やすと、より多くのトリガーによって Broker から一度にドキュメントを抽出できるのでドキュメント抽出の速度が速くなります。

ドキュメント抽出に使用されるサーバスレッド数の設定方法の詳細については、898 ページの「webMethods Messaging Triggerに使用されるサーバスレッド数の制限」を参照してください。

注:
ドキュメント抽出に割り当てられたスレッドは、Broker からドキュメントを受信する webMethods messaging trigger のみに影響します。ドキュメント抽出に指定するスレッドは、Universal Messaging からドキュメントを抽出するために使用されるスレッドを含みません。

webMethods Brokerからのドキュメント抽出に使用するスレッドの増減のタイミングについて

Broker からのドキュメント抽出に使用するスレッド数を増加または減少させるタイミングを的確に判断するには、統合ソリューションについて十分理解することが重要です。たとえば、Integration Server が通常、特定の時間帯に多数の HTTP 要求を受信することがわかっていれば、HTTP 要求が増加する直前にドキュメント抽出に使用するスレッドの数を減らし、HTTP 要求の頻度が低下してからドキュメント抽出に使用するスレッドの数を増やすことができます。または、Integration Server が毎日同じ時間帯に大量のドキュメントを受信するとわかっていれば、その時間帯のドキュメント抽出に使用可能なスレッド数を増やすことができます。

使用可能なサーバスレッドの数を監視することによって、Broker からのドキュメント抽出用スレッドを増減するタイミングを判断することもできます。そのために、使用可能なスレッドのパーセンテージが指定のレベルを下回ったときに Integration Server が通知するように警告しきい値を設定することができます。具体的には、Integration Server は「`Available Thread Warning Threshold Exceeded`」という内容のジャーナルログエントリを作成します。このメッセージをジャーナルログで受け取ったときには、ドキュメント抽出用のスレッドを減らして、使用可能なサーバスレッドを他の機能の実行に振り向けることができます。使用可能なスレッド警告しきい値の設定の詳細については、127 ページの「サーバスレッドプールの管理」を参照してください。

Broker からのドキュメント抽出に割り当てられたサーバスレッドの数を変更するタイミングを判断する別の方法として、Broker からドキュメントを抽出中の現在のスレッド数を監視するという方法もあります。Integration Server Administrator では、この値は [メッセージング] > [webMethods トリガー] ページの [ドキュメント抽出] の [現在のスレッド] フィールドに表示されます。[**ドキュメント抽出**] の [**現在のスレッド**] フィールドは、Universal Messaging からドキュメントを抽出するために使用されるスレッドが含まれません。

注:
ドキュメント抽出用のリソースを制限するその他の方法として、トリガーキューの容量を調整したり、webMethods messaging triggerのドキュメント抽出を一時停止または再開するという方法もあります。トリガーキュー容量の調整の詳細については、883 ページの「トリガーキューの容量の削減」を参照してください。ドキュメント処理の一時停止または再開の詳細については、885 ページの「ドキュメント抽出の一時停止と再開」を参照してください。

トリガーキューの容量の削減

Broker または Universal Messaging でドキュメントを抽出するすべての webMethods messaging trigger のキャパシティを調整することによって、ドキュメント抽出時のメモリ使用量に影響を与えることができます。ここでいう「キャパシティ」とは、Integration Server のトリガーキューに格納できるドキュメントの最大数のことです。

Integration Server Administrator で提供されている [**キュー容量のスロットル**] を使用して、メッセージングプロバイダからドキュメントを受信する webMethods messaging trigger のすべてのトリガーキューのキャパシティを減らすことができます。またスロットルは、Broker からドキュメントを受信するすべての webMethods messaging trigger の補充レベルを削減します。「補充レベル」とは、Integration Server が追加のドキュメントを要求する前にドキュメントキューに残存しているドキュメント数を指します。[**キュー容量のスロットル**] では、すべてのトリガーキューの容量および補充レベルが同じパーセンテージで削減されます。たとえば、[**キュー容量のスロットル**] を最大値の 50% に設定すると、容量が 10 で補充レベルが 4 のトリガーキューは、容量が 5、補充レベルが 2 に調整されます。

注:
Universal Messaging からドキュメントを受信する webMethods messaging triggerには、補充レベルがありません。

容量および補充レベルを減らすことによって、以下の結果が得られます。

- メッセージングプロバイダからのドキュメント抽出に必要なメモリ量が少なくなる。

  容量および補充レベルを減らすと、Integration Server が各 webMethods messaging trigger に対して一度に抽出するドキュメント数が少なくなります。Integration Server が抽出するドキュメント数が少なくなるので、ドキュメント抽出時に Integration Server が使用するメモリが少なくなります。

- 処理待機中のドキュメント保管に必要なメモリが少なくなる。

  注:
  容量を減らすと、Integration Server によってトリガーキューがより早く調整後の補充レベルまで空にされるため、Integration Server でドキュメントを抽出する頻度が高くなる場合があります。

webMethods Messaging Triggerの容量および補充レベルの削減

[**キュー容量のスロットル**] を利用して、すべての webMethods messaging triggerのキューの容量および補充レベルを削減するには、以下の手順に従います。

注:
[**キュー容量のスロットル**] は、Universal Messaging からドキュメントを受信する webMethods messaging triggerのトリガーが補充レベルを持たないため、補充レベルには影響しません。

トリガーキューの容量および補充レベルを減らすには

1. Integration Server Administrator を開きます。

2. [**メッセージング**] > [webMethods トリガー] に移動します。

3. [**グローバルな webMethods Messaging Triggerコントロールの編集**] をクリックします。

4. [**ドキュメント抽出**] の [**キュー容量のスロットル**] フィールドで、設定済みの容量に対するパーセンテージを選択します。ここで選択したパーセンテージはすべてのトリガーキューに一律に適用されます。補充レベルは、Integration Server によってこれと同じパーセンテージで自動的に調整されます。

5. キュー容量のスロットルの変更をクラスタのすべてのサーバに適用するには、[**クラスタ全体に変更内容を適用**] チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、webMethods messaging triggerの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。

6. [**変更内容の保存**] をクリックします。

メモ:

- キュー容量のスロットルは、Broker からドキュメントを受信する webMethods messaging triggerの補充レベルにのみ影響します。Universal Messaging からドキュメントを受信する webMethods messaging triggerには、補充レベルがありません。
- [**キュー容量のスロットル**] の設定は、サーバの再起動やパッケージの再ロードを行っても維持されます。
- 指定したパーセンテージで減少した結果、容量が整数にならない場合は、Integration Server によって端数の切り上げまたは切り捨てが行われ、最も近い整数に調整されます。ただし、切り捨ての結果、値が 0 に設定される場合は、Integration Server によって 1 に切り上げられます。たとえば、[**キュー容量のスロットル**] を最大値の 10% に設定すると、容量が 15 で補充レベルが 4 のトリガーキューは、容量が 2、補充レベルが 1 に調整されます (Integration Server によって、容量の計算結果が 1.5 から 2 に、補充レベルの計算結果が 0.4 から 1 にそれぞれ切り上げられます)。
- [**キュー容量のスロットル**] の値を減らして変更内容を保存しても、Integration Server はトリガーキュー内のドキュメント数をすぐには減少させません。調整後の補充レベルに達するまで、Integration Server はトリガーキューにあるドキュメントの処理を続行します。その上で、トリガーキューが調整後の容量に達するまで、Integration Server はドキュメントを抽出します。たとえば、[**キュー容量のスロットル**] を最大値の 50% に設定すると、容量が 8 で補充レベルが 2 のトリガーキューは、容量が 4、補充レベルが 1 に調整されます。ドキュメント数が調整後の補充レベルである 1 に達するまで、Integration Server はトリガーキューにあるドキュメントの処理を続行します。その後、Integration Server は 3 つのドキュメントを抽出して、キューのドキュメント数を 4 (調整後の容量) にします。
- 長期間にわたって容量を低いパーセンテージに減少させると、メッセージングプロバイダ上でドキュメントの有効期限が切れることがあります。パブリッシュ可能な各ドキュメントタ

イプに対して、[**廃棄**] プロパティを指定することができます。このプロパティは、メッセージングプロバイダによって廃棄されるまでドキュメントがメッセージングプロバイダに存続できる時間を指定します。パブリッシュ可能なドキュメントタイプの詳細については、『*webMethods Service Development Help*』を参照してください。

- キャパシティプランニングプロセスの一環として [**キュー容量のスロットル**] を使用し、トリガーの容量および補充レベルの設定値を変更する必要があると判断した場合は、Integration Server Administrator または Software AG Designer を使用して、各 webMethods messaging triggerに対する容量および補充レベルの値を新たに設定します。トリガーに対する容量および補充レベルを設定する場合の詳細については、903 ページの「webMethods Messaging Triggerプロパティの変更」を参照してください。

ドキュメント抽出の一時停止と再開

1 つまたは複数の webMethods messaging triggerに対するドキュメント抽出を一時停止することによって、ドキュメント抽出で消費されるサーバリソースの量を削減することができます。Integration Server Administrator では次のようなことができます。

- すべての webMethods messaging triggerに対するドキュメント抽出を一時停止または再開する。
- 特定の webMethods messaging triggerに対するドキュメント抽出を一時停止または再開する。

以下の各項で、これらのオプションの詳細について説明します。

すべてのトリガーに対するドキュメント抽出の一時停止と再開について

Integration Server 上のすべてのwebMethods messaging trigger のドキュメント抽出を一時停止すると、Integration Server は Broker および Universal Messaging からのドキュメント抽出を停止します。ドキュメント抽出に振り向けられていたスレッドやメモリなどの Integration Server リソースをその他のタスクに回せるようになります。

ドキュメント抽出をすべての webMethods messaging triggerに対して一時停止すると、サーバリソースをすばやく解放することができます。この方法は、Integration Server が高負荷で動作していて、追加のリソースをすぐに必要とする場合に特に有効です。

ドキュメント抽出の一時停止または再開は、一時的な変更とすることも永続的な変更とすることもできます(ドキュメント抽出の状態を変更するときに [**変更内容を永久に適用**] チェックボックスをオンにすると、Integration Server はこの変更を永続的と見なします)。一時的な変更の場合、Integration Server を再起動したりパッケージを再ロードしたりしたときに、Integration Server は永続的なドキュメント抽出の状態に復帰します。パッケージを再ロードした場合、Integration Server はそのパッケージに含まれる webMethods messaging triggerのみ永続的なドキュメント抽出の状態に復帰させます。たとえば、すべてのwebMethods messaging triggerのドキュメント抽出を一時的に停止した場合、OrderProcessing パッケージを再ロードすると、Integration Server は、OrderProcessing パッケージに含まれる webMethods messaging triggerに対してのみドキュメント抽出を再開します。

ヒント:

[**メッセージング**] > [webMethods トリガー] ページの [ドキュメント抽出] で、 [アクティブ] 列のトリガーの状態の横にアスタリスク (*) が表示されている場合は、ドキュメント抽出の状態が一時的な変更であることが Integration Server Administrator によって示されています。

[**キュー容量のスロットル**] を 10% 程度の低いパーセンテージに設定した上ですべての webMethods messaging triggerのドキュメント抽出を再開すると、ドキュメント抽出を徐々に再開できます。その場合、Integration Server は調整後の容量ですべての webMethods messaging triggerのドキュメント抽出を再開します。また、再開するトリガーを個別に選択することによって、ドキュメント抽出を徐々に再開することもできます。たとえば、重要なプロセスや優先順位の高いプロセスの一部であるトリガーのドキュメント抽出をまず先に再開することもできます。

ドキュメント抽出の一時停止または再開を行う前に、以下の情報に留意してください。

- ドキュメント抽出の一時停止は、メッセージングプロバイダのみ (Broker および Universal Messaging) からのドキュメント抽出にのみ影響します。webMethods messaging trigger は、ローカルでパブリッシュされるドキュメントを引き続き受信します。さらに webMethods messaging triggerは、デフォルトクライアントにデリバーされるドキュメントを引き続き受信します。
- ドキュメント抽出を一時停止すると、webMethods messaging trigger がサブスクライブするドキュメントはメッセージングプロバイダ上で収集されます。そのトリガーのドキュメント抽出が再開されるかドキュメントが期限切れになるまで、メッセージングプロバイダにドキュメントは保持されます。

  具体的には、次のいずれかが実行されます。

  - Broker からドキュメントを受信する webMethods messaging trigger の場合、Integration Server はドキュメントのポーリングを停止します。
  - Universal Messaging からドキュメントを受信する webMethods messaging trigger の場合、Integration Server はサブスクリプションを一時停止します。

    注:
    バージョン 10.7 より前の Integration Server で、Universal Messaging からメッセージを受信したトリガーを一時停止すると、Integration Server はサブスクリプションを停止し閉じます。Integration Server では、トリガーキューの未処理ドキュメントおよび現在処理中のドキュメントを含め、抽出されたが Universal Messaging に確認されていないすべてのドキュメントをロールバックしました。これは重複につながりました。サブスクリプションの一時停止によってドキュメントはロールバックされないため、重複が発生します。トリガーが有効であり、処理が一時停止されていない限り、トリガーが既に受信したメッセージは処理されます。
- webMethods messaging trigger のドキュメント抽出を一時停止するが、トリガーに対するドキュメント処理は一時停止しない場合、トリガーは既に抽出したドキュメントの処理を続行します。これには、ドキュメントをメッセージングプロバイダに確認することも含まれます。
- ドキュメント抽出を再開すると、Integration Server は、[**キュー容量のスロットル**] で指定されているパーセンテージで、webMethods messaging triggerに対するドキュメント抽出を再開します。

- ドキュメント抽出を再開しないまま Integration Server を再起動したり、トリガーパッケージを再ロードしたり、またはトリガーのプロパティを変更したりした場合、メッセージングプロバイダがトリガーのために保持している揮発性ドキュメントが廃棄されます。

すべての webMethods Messaging Triggerに対するドキュメント抽出の一時停止と再開

webMethods messaging triggerのドキュメント抽出を一時停止または再開するには、以下の手順に従います。

すべての webMethods messaging triggerに対するドキュメント抽出を一時停止または再開するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。
3. [個別の **webMethods Messaging Trigger** 制御] セクションで、[ドキュメント抽出] にある [アクティブ] 列の [一括編集] をクリックします。
4. [**抽出の状態**] リストで、次のいずれかを選択します。

選択項目	目的
[アクティブ]	すべての webMethods messaging triggerに対するドキュメント抽出を再開する。
[一時停止]	すべての webMethods messaging triggerに対するドキュメント抽出を一時停止する。

5. 状態の変更を永続的にして Integration Server の再起動後またはパッケージの再ロード後にも変更が維持されるようにするには、[**変更内容を永久に適用**] チェックボックスをオンにします。このチェックボックスをオンにしない場合、Integration Server は一時的な変更と見なします。
6. ドキュメント抽出の変更をクラスタのすべてのサーバに適用するには、[**クラスタ全体に変更内容を適用**] チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。
7. [**変更内容の保存**] をクリックします。

メモ:

- webMethods messaging trigger がロックされているか無効である場合は、Integration Server はドキュメント抽出を一時停止または再開しません。
- Integration Server がローカルの Integration Server でドキュメント抽出を一時停止または再開できない場合、クラスタ同期は発生しません。
- `watt.server.trigger.managementUI.excludeList` を使用してトリガー管理の変更から除外している webMethods messaging trigger に対して、Integration Server はドキュメント抽出を一時停止または再開しません。このプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

特定のトリガーに対するドキュメント抽出の一時停止と再開について

すべての webMethods messaging trigger のドキュメント抽出を一時停止または再開するのではなく、特定の webMethods messaging trigger のドキュメント抽出を一時停止または再開する方がよい場合もあります。たとえば、次のような場合は、特定のトリガーのドキュメント抽出を一時停止または再開することをお勧めします。

- バックエンドシステムの保守が必要な場合や応答が鈍ってきている場合は、バックエンドシステムとやりとりしているトリガーのドキュメント抽出を一時停止します。ドキュメント抽出を一時停止することによって、処理の待機のために通常は Integration Server に蓄積されるドキュメントがメッセージングプロバイダに保持されます。この動作によって、メモリやその他のサーバリソースを他のアクティビティで使用できます。バックエンドシステムが使用可能になったら、該当トリガーのドキュメント抽出を再開します。
- すべてのトリガーのドキュメント抽出を一時停止した上で、特定のトリガーのドキュメント抽出を再開するとよい場合があります。Integration Server が通常にはない重い負荷で動作している場合は、まずすべてのトリガーのドキュメント抽出を一時停止し、その上で重要なプロセスに関わるトリガーから順に少しずつ抽出を再開していくことができます。
- 関連付けられたトリガーサービスがエラーで終了したために、Integration Server によって逐次実行トリガーに対するドキュメント抽出が一時停止されている場合、そのトリガーに対するドキュメント抽出を再開する必要があります。エラー後に抽出および処理を自動的に一時停止するための逐次実行トリガーの設定の詳細については、『*webMethods Service Development Help*』を参照してください。

特定の webMethods Messaging Triggerに対するドキュメント抽出の一時停止と再開

個別の webMethods messaging triggerのドキュメント抽出を一時停止または再開するには、以下の手順に従います。

特定の webMethods messaging triggerに対するドキュメント抽出を一時停止または再開するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。

3. [個別の **webMethods Messaging Trigger** 制御] で、ドキュメント抽出を一時停止するトリガーの行を探します。

4. [**ドキュメント抽出**] の下にある [**アクティブ**] 列で、[**はい**] または [**いいえ**] をクリックします。(トリガーのドキュメント抽出がアクティブである場合、[**アクティブ**] 列には [はい] が表示されます。ドキュメント抽出が一時停止されている場合、[いいえ] が表示されます。ドキュメント抽出の状態が一時的である場合、状態の隣にアスタリスク (*) が表示されます)。

5. [**抽出の状態**] リストで、次のいずれかを選択します。

選択項目	目的
[**アクティブ**]	このトリガーのドキュメント抽出を再開します。
[**一時停止**]	このトリガーのドキュメント抽出を一時停止します。

6. 状態の変更を永続的にして Integration Server の再起動後も変更が維持されるようにするには、[**変更内容を永久に適用**] チェックボックスをオンにします。このチェックボックスをオンにしない場合、Integration Server は一時的な変更と見なします。

7. このトリガーのドキュメント抽出の変更をクラスタのすべてのサーバに適用するには、[**クラスタ全体に変更内容を適用**] チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。

8. [**変更内容の保存**] をクリックします。

メモ:

- 指定した webMethods messaging triggerがユーザによってロックされている場合、Integration Server はそのトリガーに対するドキュメント抽出を一時停止または再開しません。
- Integration Server がドキュメント抽出をローカルで一時停止または再開できない場合、クラスタ同期は発生しません。
- ドキュメント抽出を再開すると、Integration Server は、[**キュー容量のスロットル**] で指定されているパーセンテージで、その webMethods messaging triggerに対するドキュメント抽出を再開します。
- フローサービスでは、pub.trigger:suspendRetrieval サービス、または pub.trigger:resumeRetrieval サービスをそれぞれ呼び出すことによって、個別のトリガーに対するドキュメント抽出を一時停止または再開することができます。これらのサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

- Java サービスでは、com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade.setRetrievalSuspended() を呼び出すことによってドキュメント処理を一時停止または再開することができます。このメソッドの詳細については、『*webMethods Integration Server Java API Reference*』の com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade クラスを参照してください。
- watt.server.trigger.managementUI.excludeList プロパティを使用することによって、表示されるトリガーのリストをフィルタすることができます。このプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

ドキュメント処理の管理

パブリッシュ/サブスクライブモデルにおいて、「ドキュメント処理」とは、ドキュメントをトリガー条件と照合して評価し、そのドキュメントに対して動作する適切なトリガーサービスを実行するプロセスをいいます。ドキュメント処理には、ドキュメントの評価とトリガーサービスの実行のためのサーバスレッドが必要です。また、ドキュメントの評価とトリガーサービスの実行の間にドキュメントのコピーを保持するためのメモリも必要です。

Integration Server には、ドキュメント処理の速度およびリソース要求を管理するためのさまざまな制御が備わっています。具体的には次の操作が可能です。

- webMethods messaging triggerがドキュメント処理に使用するサーバスレッド数を制限する。
- webMethods messaging triggerに対してドキュメントを並行処理するために使用されるサーバスレッドの数を管理する。
- 1 つまたは複数の webMethods messaging triggerに対してドキュメント処理を一時停止または再開する。

これらの制御は、キャパシティプランニングの一環としてまたは実稼動環境で使用できます。以下の項目では、これらの制御について詳しく説明します。

ドキュメントの処理で webMethods Messaging Triggerが受信するスレッド数の増減

実稼動環境やキャパシティプランニングで、webMethods messaging trigger によって受信されるドキュメントの並行処理に使用できるスレッド数を増加または減少させることができます。ドキュメントの処理に使用可能なスレッドの数は、Integration Server がドキュメントを処理する速度を決定する要因になります。デフォルトでは、Integration Server はサーバスレッドプールを最大で 100% 使用してドキュメントを処理 (トリガーを実行) します。Integration Server がドキュメントを処理するたびに、Integration Server はサーバスレッドを使用します。たとえば、サーバスレッドプールの最大サイズが 80 スレッドで、Integration Server がサーバスレッドプールを 100% 使用してトリガーを実行できる場合、最大 80 のトリガーを同時に実行することができます。つまり、Integration Server は最大 80 のドキュメントを同時に処理することができます。

ドキュメント処理に使用できるサーバスレッドプールの最大値に対するパーセンテージを指定することによって、ドキュメント処理 (トリガーの実行) で使用可能なサーバスレッド数を制御できます。Integration Server は、このパーセンテージを使用して、トリガーの実行 (ドキュメント

処理) に使用するサーバスレッド数を計算します。たとえば、サーバスレッドプールの最大サイズが 80 スレッドであるとします。ドキュメント処理のスレッドの最大値に対するパーセンテージを 10% に指定すると、Integration Server は最大 8 個のスレッドを使用して webMethods messaging trigger を同時に実行できます。

ドキュメント処理に使用されるサーバスレッド数を減らすことによって、HTTP 要求の実行やドキュメント抽出など、他のタスクの実行にサーバスレッドやメモリを使用できます。または、ドキュメント処理用のスレッドの数を増やすことによって、サーバがより多くのドキュメントを同時に処理できるように設定できます。この場合、サーバはトリガーキューをよりすばやく排出するので、より頻繁に後続のドキュメントを要求できます。

逐次実行トリガーと同時実行トリガーの組み合わせに対するドキュメント処理は、ドキュメント処理スレッドの最大値に対するパーセンテージによって決定される値を超えることはできません。ドキュメント処理のサーバスレッドのパーセンテージを減らした場合、指定された設定に基づいて使用可能な最大数の実行スレッドを同時実行トリガーが使用し続けると、サーバスレッドが使用可能になるまで逐次実行トリガーが待機する必要がある時間が長くなります。この状況は、Integration Server に長時間にわたるサービスを実行する同時実行トリガーがある場合に発生する可能性が特に高くなります。

ドキュメント処理に使用されるサーバスレッド数を設定する方法の詳細については、898 ページの「webMethods Messaging Triggerに使用されるサーバスレッド数の制限」を参照してください。

ヒント:
ドキュメント処理に使用できるスレッドのパーセンテージを減らす場合、[**実行スレッドのスロットル**] の値を下げて使用可能なサーバスレッドを同時実行トリガーが独占しないようにすることを検討してください。

ドキュメント処理に使用するスレッドの増減のタイミングについて

ドキュメント処理 (トリガーの実行) に使用するスレッド数を調整するタイミングを的確に判断するには、統合ソリューションについて十分理解することが重要です。たとえば、毎日同じ時間に発生するバッチプロセスによって、ドキュメントのパブリッシュが急増するとします。この場合、バッチプロセスが開始する直前にドキュメント処理に使用するスレッド数を増やして、サーバスレッドをドキュメント処理に使用できるようにすることができます。

または、メモリの制約やその他のリソースの問題がある場合は、ドキュメント処理に使用するスレッド数を減らすことができます。サーバスレッドでドキュメントを評価しトリガーサービスを実行している間、Integration Server はドキュメントをメモリに保持するため、ドキュメント処理によってメモリが消費されます。

注:
スレッドの使用状況を監視することによって、ドキュメント処理に許可するスレッド数を変更するタイミングを判断することもできます。そのためには、[**サーバ**] > [統計情報] ページに表示されるスレッド使用状況の情報を参照します。さらに、使用可能なスレッド数が特定レベルより下になったときに Integration Server が通知するように警告しきい値を設定することもできます。具体的には、Integration Server は「`Available Threads Warning Threshold Usage Exceeded`」という内容のジャーナルログエントリを作成します。Integration Server がこのジャーナルログエントリを書き込んだ場合、ドキュメント処理のスレッドを減らして、その他の機能でより多くのスレッドを使用できるようにできます。使用可能なスレッド警告しき

い値の設定の詳細については、127 ページの「サーバスレッドプールの管理」を参照してください。

ドキュメント処理に割り当てられたサーバスレッドの数を変更するタイミングを判断する別の方法として、トリガーに対してドキュメントを処理しているスレッドの現在の数を監視します。Integration Server Administrator では、この値は [**メッセージング**] > [**webMethods トリガー**] ページの [ドキュメント処理] にある [現在のスレッド] フィールドに表示されます。

注:
ドキュメント処理に使用されるリソースを制限するその他の方法として、同時実行トリガーの実行スレッドを調整したり、トリガーのドキュメント処理を一時停止または再開するという方法もあります。トリガーキュー容量の調整の詳細については、892 ページの「同時実行トリガーのドキュメント処理量の削減」を参照してください。ドキュメント処理の一時停止または再開の詳細については、894 ページの「ドキュメント処理の一時停止と再開」を参照してください。

同時実行トリガーのドキュメント処理量の削減

同時実行トリガーに対する処理速度を下げることによって、ドキュメント処理で消費されるサーバリソースの量を削減することができます。具体的には、同時実行トリガーに対して一度にドキュメントを処理できるスレッドの最大数を減らします。

Integration Server Administrator に用意されている [**実行スレッドのスロットル**] を使用すると、すべての同時実行トリガーに対する実行スレッドを同じパーセンテージで削減することができます。たとえば、[**実行スレッドのスロットル**] を最大値の 50% に設定すると、Integration Server はすべての同時実行トリガーに対する実行スレッドの最大数を半分に削減します。実行スレッドの最大数の値が 6 である同時実行トリガーは 3 に調整されます。

同時実行トリガーの並行処理量を減少させることによって、以下の結果が得られます。

- サーバスレッドとメモリを解放して、HTTP 要求への応答やドキュメント抽出などの他の機能を実行する。
- ドキュメント処理に割り当てられたスレッドを同時実行トリガーが独占しないようにする。逐次実行トリガーおよび同時実行トリガーに対するドキュメント処理のために Integration Server がディスパッチするサーバスレッド数は、実行スレッドの最大数に対するパーセンテージで設定されている値を超えることはできません。ドキュメント処理に使用できるスレッド数を減らした場合、指定された設定に基づいて使用可能な最大数の実行スレッドを同時実行トリガーが使用し続けると、サーバスレッドが使用可能になるまで逐次実行トリガーが待機する必要がある時間が長くなります。この状況は、Integration Server に長時間にわたるサービスを実行する同時実行トリガーがある場合に発生する可能性が特に高くなります。

注:
[**実行スレッドのスロットル**] は、webMethods messaging triggerにのみ影響します。

同時実行 webMethods Messaging Triggerに対する実行スレッドの削減

同時実行 webMethods messaging triggerに対する実行スレッドを減らすには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。
3. [**グローバルな webMethods Messaging Triggerコントロールの編集**] をクリックします。
4. [**ドキュメント処理**] の [**実行スレッドのスロットル**] フィールドで、実行スレッドの最大数の設定値に対してサーバが動作するパーセンテージを選択します。Integration Server によって、このパーセンテージは、すべての同時実行 webMethods messaging triggerの実行スレッドの最大数の値に適用されます。
5. [**実行スレッドのスロットル**] の変更をクラスタのすべてのサーバに適用するには、[**クラスタ全体に変更内容を適用**] チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。
6. [**変更内容の保存**] をクリックします。

メモ:

- [**実行スレッドのスロットル**] の値は、サーバの再起動やパッケージの再ロードを行っても維持されます。
- 逐次実行トリガーは、常に一度に 1 つのドキュメントを処理します。[**実行スレッドのスロットル**] プロパティは、逐次実行トリガーに影響しません。
- 指定したパーセンテージで減少した結果、トリガーの実行スレッド数が整数にならない場合は、Integration Server によって端数の切り上げまたは切り捨てが行われ、最も近い整数に調整されます。ただし、切り捨ての結果、値が 0 に設定される場合は、Integration Server によって 1 に切り上げられます。たとえば、[**実行スレッドのスロットル**] を最大値の 10% に減少させた場合、実行スレッドの最大数の値が 12 に設定されている同時実行トリガーの値は 1 に調整されます (Integration Server によって 1.2 が 1 に切り捨てられます)。実行スレッドの最大数の値が 4 に設定されている同時実行トリガーの値は 1 に調整されます (Integration Server によって 0.4 が 1 に切り上げられます)。
- [**実行スレッドのスロットル**] の値を減らして変更内容を保存しても、Integration Server は現在同時実行トリガーを実行しているスレッドを終了して、調整された最大値に合わせることはしません。Integration Server では、同時実行トリガーのドキュメントを処理するサーバスレッドは、処理を完了するまで実行されます。Integration Server は、同時実行トリガーの処理を実行しているスレッドの数が、調整された最大値を下回るまで待機してから、次のサーバスレッドをディスパッチして、そのトリガーのドキュメントを処理します。
- ドキュメント処理 (トリガーの実行) を一時停止した場合は、ドキュメント抽出を一時停止しないと、Integration Server によってすべてのトリガーキューが容量いっぱいまで満たされま

す。トリガーキューがいっぱいになると、トリガーキューが空の場合よりも、メモリの消費量が多くなります。

- トリガーキューの容量がトリガーに対する同時実行スレッドの最大数を下回るように設定することによって、トリガーに対する同時実行スレッド数を減らすこともできます。トリガーに対してディスパッチされるスレッドの最大数は、トリガーキューの容量を超えることはできません。トリガーキュー容量の削減の詳細については、883 ページの「トリガーキューの容量の削減」を参照してください。
- キャパシティプランニングプロセスの一環として [**実行スレッドのスロットル**] を使用し、[**実行スレッドの最大数**] の設定値を変更する必要があると判断した場合は、Integration Server Administrator または Software AG Designer を使用して、各同時実行トリガーに対する実行スレッドの最大数の値を新たに設定してください。トリガーのプロパティの設定の詳細については、903 ページの「webMethods Messaging Triggerプロパティの変更」を参照してください。

ドキュメント処理の一時停止と再開

1 つまたは複数の webMethods messaging triggerに対するドキュメント処理を一時停止することによって、ドキュメント処理で消費されるサーバリソースの量を削減することができます。Integration Server Administrator では次のようなことができます。

- すべての webMethods messaging trigger のドキュメント処理を一時停止または再開する。
- 特定の webMethods messaging trigger のドキュメント処理を一時停止または再開する。

以下の各項で、これらのオプションの詳細について説明します。

すべてのトリガーに対するドキュメント処理の一時停止と再開について

すべての webMethods messaging trigger に対してドキュメント処理を一時停止すると、Integration Server は webMethods messaging trigger によって受信されるドキュメントを処理するためのサーバスレッドのディスパッチを停止します。ドキュメント処理のために使用されていたスレッド、メモリなどのサーバリソースが、その他のタスクで使用可能になります。ドキュメント処理は、明示的に再開するまで一時停止のままです。

ドキュメント処理をすべての webMethods messaging triggerに対して一時停止すると、サーバリソースをすばやく使用可能にすることができます。この方法は、Integration Server が高負荷で動作していて、追加のリソースをすぐに使用可能にする必要がある場合に特に有効です。

ドキュメント処理の一時停止または再開は、一時的な変更または永続的な変更とすることができます(ドキュメント処理の状態を変更するときに [**変更内容を永久に適用**] チェックボックスをオンにすると、Integration Server はこの変更を永続的と見なします)。一時的な変更の場合、Integration Server を再起動したりパッケージを再ロードしたりしたときに、Integration Server は永続的なドキュメント処理の状態に復帰します。パッケージを再ロードした場合、Integration Server は、そのパッケージに含まれるトリガーのみ永続的なドキュメント処理の状態に復帰させます。たとえば、すべてのトリガーのドキュメント処理を一時的に停止した場合、パッケージ OrderProcessing を再ロードすると、Integration Server は、OrderProcessing パッケージに含まれるトリガーに対してのみドキュメント処理を再開します。

ヒント:

[**メッセージング**] > [webMethods トリガー] ページの [ドキュメント処理] で、[アクティブ] 列のトリガーの状態の横にアスタリスク (*) が表示されている場合は、ドキュメント処理の状態が一時的な変更であることが Integration Server Administrator によって示されています。

ドキュメント処理の再開は徐々に行うことができます。たとえば、[**実行スレッドのスロットル**] を低いパーセンテージに設定してすべてのトリガーに対するドキュメント処理を再開し、その後徐々に [**実行スレッドのスロットル**] を 100% まで上げます。または、トリガーを個別に選択して再開することもできます。たとえば、重要なプロセスや優先順位の高いプロセスの一部であるトリガーのドキュメント処理を先に再開することもできます。

すべての webMethods Messaging Triggerに対するドキュメント処理の一時停止と再開

すべての webMethods messaging triggerのドキュメント処理を一時停止または再開するには、以下の手順に従います。

すべての webMethods messaging triggerに対するドキュメント処理を一時停止または再開するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。
3. [**個別 webMethods Messaging Trigger 制御**] セクションで、[**ドキュメント処理**] にある [**アクティブ**] 列の [**一括編集**] をクリックします。
4. [**処理の状態**] リストで、次のいずれかを選択します。

選択項目	目的
[**アクティブ**]	すべての webMethods messaging triggerのドキュメント処理を再開します。
[**一時停止**]	すべての webMethods messaging triggerのドキュメント処理を一時停止します。

5. 状態の変更を永続的にして Integration Server の再起動後も変更が維持されるようにするには、[**変更内容を永久に適用**] チェックボックスをオンにします。このチェックボックスをオンにしない場合、Integration Server は一時的な変更と見なします。
6. ドキュメント処理の変更をクラスタのすべてのサーバに適用するには、[**クラスタ全体に変更内容を適用**] チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。

7. [**変更内容の保存**] をクリックします。

メモ:

- トリガーがロックされているか無効である場合、Integration Server はドキュメント処理を一時停止または再開しません。
- Integration Server がドキュメント処理をローカルで一時停止または再開できない場合、クラスタ同期は発生しません。
- `watt.server.trigger.managementUI.excludeList` を使用してトリガー管理の変更から除外している webMethods messaging triggerに対して、Integration Server はドキュメント処理を一時停止または再開しません。このプロパティの詳細については、以下を参照してください。 1095 ページの「サーバ設定パラメータ」
- ドキュメント処理の一時停止または再開の影響は、メッセージングプロバイダから抽出されたドキュメントやローカルパブリッシュからのドキュメントを含めて、Integration Server 上のトリガーのキューにあるすべてのドキュメントに及びます。
- ドキュメント処理を一時停止すると、Integration Server はドキュメントを処理するサーバスレッドをそれ以上ディスパッチしません。現在トリガーに対してドキュメントを処理しているサーバスレッドは、いずれも処理完了まで実行されます。再試行中のドキュメント処理も完了まで実行されます。
- ドキュメント処理を一時停止した場合は、ドキュメント抽出を一時停止しないと、トリガーキューが最大容量に達するまで、またはドキュメント処理が再開するまでドキュメントがトリガーキューに収集されます。ドキュメント処理の再開の前に Integration Server が再起動すると、揮発性ドキュメントは廃棄されます。
- ドキュメント処理を再開すると、Integration Server は、[**実行スレッドのスロットル**] で指定されているパーセンテージで webMethods messaging triggerのドキュメント処理を再開します。

特定のトリガーに対するドキュメント処理の一時停止と再開について

すべての webMethods messaging trigger のドキュメント処理を一時停止または再開するのではなく、特定のトリガーのドキュメント処理を一時停止または再開する方がよい場合もあります。たとえば、次のような場合は、特定のトリガーのドキュメント処理を一時停止または再開することをお勧めします。

- バックエンドシステムの応答が鈍ってきている場合や保守が必要な場合は、バックエンドシステムとやりとりしているトリガーのドキュメント処理を一時停止します。保守または障害のためにバックエンドシステムが使用できないときには、多くの場合、バックエンドシステムとやりとりするトリガーサービスは正常に実行されません。関連するトリガーのドキュメント処理を一時停止すると、失敗したドキュメント処理に使用されていたリソースを他のタスクに使用できるため、リソースをより効果的に使用することができます。
- すべてのトリガーのドキュメント処理を一時停止した上で、特定のトリガーのドキュメント処理を再開するとよい場合があります。Integration Server が重い負荷で動作している場合、最初にすべてのドキュメント処理を一時停止し、次に重要なプロセスに関連するトリガーから始めて徐々にドキュメント処理を再開します。

- 関連付けられたトリガーサービスがエラーで終了したために、Integration Server が逐次実行トリガーに対するドキュメント処理を一時停止した場合、そのトリガーに対するドキュメント処理を再開する必要があります。エラー後に抽出および処理を自動的に一時停止するための逐次実行トリガーの設定の詳細については、『*webMethods Service Development Help*』を参照してください。

特定の webMethods Messaging Triggerに対するドキュメント処理の一時停止と再開

個別の webMethods messaging triggerのドキュメント処理を一時停止または再開するには、以下の手順に従います。

特定の webMethods messaging triggerに対するドキュメント抽出を一時停止または再開するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。
3. [**個別 webMethods Messaging Trigger 制御**] で、ドキュメント処理を一時停止するトリガーの行を探します。
4. [**ドキュメント処理**] の下にある [**アクティブ**] 列で、[**はい**] または [**いいえ**] をクリックします。(トリガーのドキュメント処理がアクティブである場合、[**アクティブ**] 列には [はい] が表示されます。ドキュメント処理が一時停止されている場合、[いいえ] が表示されます。ドキュメント処理の状態が一時的である場合、状態の隣にアスタリスク (*) が表示されます)。
5. [**処理の状態**] リストで、次のいずれかを選択します。

選択項目	目的
[**アクティブ**]	この webMethods messaging triggerのドキュメント処理を再開します。
[**一時停止**]	この webMethods messaging triggerのドキュメント処理を一時停止します。

6. 状態の変更を永続的にして Integration Server の再起動後も変更が維持されるようにするには、[**変更内容を永久に適用**] チェックボックスをオンにします。このチェックボックスをオンにしない場合、Integration Server は一時的な変更と見なします。
7. このトリガーのドキュメント処理の変更をクラスタのすべてのサーバに適用するには、[**クラスタ全体に変更内容を適用**] チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。

8. [**変更内容の保存**] をクリックします。

メモ:

- 指定した webMethods messaging triggerがユーザによってロックされている場合、Integration Server はそのトリガーに対するドキュメント処理を一時停止または再開しません。
- Integration Server がドキュメント処理をローカルで一時停止または再開できない場合、クラスタ同期は発生しません。
- ドキュメント処理を再開した場合、並行処理できるドキュメントの最大数は [**実行スレッドのスロットル**] によって決まります。
- フローサービスでは、pub.trigger:suspendProcessing サービス、または pub.trigger:resumeProcessing サービスをそれぞれ呼び出すことによって、個別の webMethods messaging trigger に対するドキュメント処理を一時停止または再開することができます。これらのサービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。
- Java サービスでは、com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade.setProcessingSuspended() を呼び出すことによってドキュメント処理を一時停止または再開することができます。このメソッドの詳細については、『*webMethods Integration Server Java API Reference*』の com.wm.app.b2b.server.dispatcher.trigger.TriggerFacade クラスを参照してください。
- watt.server.trigger.managementUI.excludeList プロパティを使用することによって、表示されるトリガーのリストをフィルタすることができます。このプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

webMethods Messaging Triggerに使用されるサーバスレッド数の制限

Integration Server には、webMethods messaging triggerのドキュメントの抽出と処理に使用されるサーバスレッドプール内のスレッド数を制限するためのパラメータが用意されています。メッセージングプロバイダからドキュメントを抽出する際に使用可能なサーバスレッドプールのパーセンテージを指定することができます。また、トリガーによって受信されるドキュメントを処理する際に使用可能なサーバスレッドプールの割合も指定できます。

たとえば、サーバスレッドプールに最大 80 個のスレッドを含めることができるとします。このとき、ドキュメント抽出の最大スレッドをサーバスレッドプールの 50% に設定すると、Integration Server ではメッセージングプロバイダからのドキュメント抽出に最大 40 個のスレッドを使用することができます。

メッセージングプロバイダからのドキュメント抽出と処理に使用するサーバスレッドの数を制限することで、サーバスレッドを他のサーバ機能 (HTTP 要求への応答、サーバ管理など) の実行に使用することができます。これらのパラメータを使用すると、メッセージングプロバイダからのドキュメントの抽出と処理でサーバスレッドプール全体を独占しないように設定できます。

注:

Integration Server がドキュメント抽出にスレッドを使用すると、そのスレッドは 1 つのトリガーに使用するドキュメントをメッセージングプロバイダから取得します。1 つのスレッドですべてのトリガーに対するドキュメントを抽出するわけではありません。ドキュメントを処理するために使用される 1 つのスレッドでは、トリガーキュー内のドキュメントが 1 つ処理されます(ドキュメントの処理には、ドキュメントが満たす必要のあるトリガー条件の決定および関連サービスの実行が含まれます)。

webMethods Messaging Triggerで使用されるサーバスレッドの最大数の設定

webMethods messaging triggerに対するドキュメントの抽出と処理に使用されるスレッドの数を指定する場合は、以下の点に留意してください。

- [ドキュメント抽出] の最大スレッドに対するパーセンテージは、Broker からドキュメントを抽出する webMethods messaging triggerにのみ影響します。
- [ドキュメント処理] の最大スレッドに対するパーセンテージは、すべての webMethods messaging triggerに影響を与えます。
- Universal Messaging からメッセージを受信する webMethods messaging trigger ごとに、Integration Server はスレッドを使用してトリガーを実行します。トリガーの実行には、トリガーの開始とメッセージの受信待機が含まれます。トリガーの実行とメッセージの受信待機に使用されるスレッドはドキュメント処理スレッドとは見なされません。つまり、トリガーを実行するスレッドは、ドキュメント処理の [**最大スレッド**] によって設定された使用上限に加算されません。トリガーの実行とメッセージの受信待機に使用されるスレッドが、ドキュメント処理の [**最大スレッド**] によって設定された使用上限に対して加算されるようにする場合は、watt.server.messaging.trigger.startup.useServerThread サーバ設定パラメータを「false」に設定します。

webMethods messaging triggerで使用されるサーバスレッドの最大数を設定するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。
3. [**グローバルな webMethods Messaging Triggerコントロールの編集**] をクリックします。
4. [**ドキュメント抽出**] の [**最大スレッド**] フィールドに、Broker からドキュメントを抽出する際に使用可能なサーバスレッドプールの最大値に対するパーセンテージを入力します。0 より大きい値を入力してください。デフォルトは 100% です。

   [ドキュメント抽出] の [**最大スレッド**] フィールドに「Broker 未設定」と表示される場合、この Integration Server は Broker に接続するように設定されていません。
5. [**ドキュメント処理**] の [**最大スレッド**] フィールドに、webMethods messaging triggerが受信するドキュメントを処理する際に使用可能なサーバスレッドプールの最大値に対するパーセンテージを入力します。0 より大きい値を入力してください。デフォルトは 100% です。

6. [**変更内容の保存**] をクリックします。

メモ:

- 入力したパーセンテージに応じて、ドキュメントの抽出と処理に使用できるスレッドの数が Integration Server によって自動的に計算されます。スレッドの数が整数にならない場合、Integration Server によって端数の切り上げまたは切り捨てが行われ、最も近い整数になります。
- ドキュメント抽出では、ドキュメントを抽出するサーバスレッドの現在の数が新たに [**最大スレッド**] のパーセンテージで設定された値よりも大きい場合、Integration Server は Broker からドキュメントの抽出に使用するスレッドをそれ以上ディスパッチしません。現在ドキュメントを抽出しているスレッドは、完了するまで実行されます。ドキュメント抽出スレッドの現在の数が、ドキュメント抽出スレッドの最大許可数より小さい場合にのみ、Integration Server はドキュメントの抽出に使用する新しいスレッドを Broker からディスパッチします。
- ドキュメント処理では、ドキュメントを処理する (トリガーを実行する) サーバスレッドの現在の数が [**最大スレッド**] で決定されたスレッドの数よりも大きい場合、Integration Server はドキュメント処理に使用するスレッドをそれ以上ディスパッチしません。現在ドキュメントを処理しているスレッドは、完了するまで実行されます。ドキュメント処理スレッドの現在の数が、ドキュメント処理スレッドの最大許可数より小さい場合にのみ、Integration Server はトリガーの実行に使用する新しいスレッドをディスパッチします。
- Broker からのドキュメント抽出用のスレッドの現在数と最大スレッドは、[メッセージング] > [webMethods トリガー] ページの [ドキュメント抽出] にある [現在のスレッド] フィールドに表示されます。
- ドキュメント処理用のスレッドの現在数と最大スレッドは、[メッセージング] > [webMethods トリガー] ページの [ドキュメント処理] にある [現在のスレッド] フィールドに表示されます。

webMethods Messaging Trigger管理のクラスタ同期

Integration Server がクラスタのメンバーである場合、ドキュメント抽出、ドキュメント処理およびトリガーのプロパティに関して行った変更は、クラスタのその他のサーバに自動的に反映されます。クラスタのその他のサーバに変更が自動的に反映されることによって、同じ変更をすべてのサーバに対して手動で行う必要はありません。さらに、最初のサーバに対して行われたリソース要求をその他のサーバが負担することもありません。

注:
pub.trigger サービスを使用して行ったトリガー管理の変更も、クラスタ全体に適用されます。

クラスタ同期の設定

Integration Server は、リモート呼び出しを実行することによって、クラスタのその他のメンバーにトリガーの管理の変更を反映します。クラスタ同期を正常に行うには、次の作業を完了する必要があります。

- クラスタの設定。すべてのサーバが、適切に設定されたクラスタのメンバーになっている場合のみ、Integration Server はトリガー管理の変更をその他のサーバに反映できます。クラスタの設定の詳細については、*webMethods Integration Server Clustering Guide*を参照してください。
- クラスタのサーバに対するリモートサーバエイリアスの設定。リモートサーバエイリアスの設定の詳細については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。
- カンマ区切りのリモートサーバエイリアス名リストを使用した watt.server.cluster.aliasList プロパティの更新。Integration Server は、リモート呼び出しを実行してその他のクラスタノードを更新する際にこのリストを使用します。

  注:
  クラスタに属するがこのリストにはない Integration Server は、クラスタ同期では更新されません。

クラスタ同期プロパティ watt.server.cluster.aliasList が適切に設定されている場合、トリガー管理タスクの実行時に [**クラスタ全体に変更内容を適用**] チェックボックスが表示されます。

実行時のクラスタ同期

実行時に、Integration Server はリモート呼び出しを使用して、クラスタの他のメンバーをトリガーの管理の変更内容により更新します。サーバに対するリモート呼び出しが失敗した場合、またはリモート呼び出し時にサーバが使用可能でなかった場合、クラスタは同期されません。リモート呼び出しを実行する Integration Server は、次のジャーナルログメッセージを表示して、クラスタ同期の試行の状態を示します。

以下の表は、Integration Server が各同期の試行の結果に対して書き込むログメッセージを示しています。

同期の状態	表示されるメッセージ
成功	Integration Server Administrator は次のメッセージを表示します。 `設定が正常に変更されました。`
成功したサーバと失敗したサーバがある	Integration Server Administrator は次のメッセージを表示します。 `[ISS.0085.9203] リモートエイリアスの更新中にエラーが発生しました (`*y* ` 件中の `*x*` 件の更新が失敗しました)。詳細については サーバログを参照してください。` サーバログには、更新に成功しなかったクラスタのメンバーごとに次のメッセージが表示されます。 `[ISS.0098.0107E] クラスタ呼び出し中にエラーが発生しました。 エイリアス = `*remoteAliasName*`、サービス = `*serviceName*`、 例外 = `*exceptionName* Integration Server Administrator を使用して、トリガーに対するクラスタ同期の状態を表示および変更できます。

同期の状態	表示されるメッセージ
ローカルの更新のエラーのために失敗	Integration Server Administrator は次のメッセージを表示します。 `[ISS.0085.9204]` **ローカルの更新が失敗しました： 失敗の原因を示す例外(注：ローカルのエラーがすべて解決されるまでクラスタの同期は実行されません)。** トリガー管理の変更がローカルの Integration Server で完了できない場合、クラスタ同期は発生しません。たとえば、すべてのトリガーのドキュメント抽出を一時停止する際に、1 つのトリガーが現在ロックされていると、Integration Server はロックされているトリガーを除くすべてのトリガーのドキュメント抽出を一時停止します。ドキュメント抽出がローカルで完了できないため、Integration Server はクラスタの他のサーバと変更を同期できません。
設定されていないために失敗	サーバログに次のメッセージが表示されます。 `[ISS.0033.0156W] クラスタ呼び出しが正常に完了しませんでした。クラスタ同期機能が設定されていません。` トリガーに対するクラスタ同期の設定の詳細については、900 ページの「クラスタ同期の設定」を参照してください。

クラスタ同期の監視

Integration Server Administrator には、クラスタ内のすべてのサーバに関してトリガーの同期状態を確認するためのクラスタの表示が用意されています。クラスタの表示では、watt.server.cluster.aliastList プロパティにリストされているサーバごとに、クラスタの他のサーバ (およびクラスタのトリガー) が現在のサーバと同期されているかどうかが示されます。クラスタの表示は、[メッセージング] > [webMethods トリガー] > [クラスタの表示] ページにあります。

注:
[**クラスタの表示**] ページが表示されるのは、現在の Integration Server が、適切に設定されたクラスタに属していて、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。

webMethods messaging triggerが同期されていない場合、クラスタの表示には、現在のサーバにあるトリガーと、トリガーが同期されていない理由を示すエラーメッセージが表示されます。たとえば、トリガーに対するドキュメント処理がローカルで一時停止されているが、クラスタの別のサーバではアクティブである場合、トリガー名の隣のエラーメッセージに「Processing State mismatch [local=suspended; remote=active]」と表示されます。

次のいずれかに該当する場合、Integration Server は、リモートサーバのトリガーが同じ名前のローカルトリガーと同期されていないと見なします。

- トリガーキューの容量、補充レベルまたはトリガーの実行スレッドの最大数の値がトリガー間で異なる。
- ドキュメント抽出またはドキュメント処理の状態がトリガー間で異なる。

注:
クラスタのリモートサーバにログオンするには、[**リモートサーバエイリアス**] 列でサーバエイリアスをクリックします。接続時に、リモートサーバによって、ユーザおよびパスワードについての情報を入力するよう要求されます。HTTPS を介してリモートサーバに接続するために HTTPS ポートに認証が必要な場合、信用のある認証をブラウザにインポートして接続時に提示されるようにします。信用のある認証をブラウザにインポートしていない場合、リモートサーバに接続しようとすると、ページを使用できないというメッセージが表示されます。クライアント認証の詳細については、549 ページの「クライアントの認証」を参照してください。

webMethods Messaging Triggerプロパティの変更

キャパシティプランニングまたは実稼動環境で、設定済みのトリガーのプロパティをリセットする必要がある場合があります。たとえば、一部のトリガーについて、そのトリガーのキューの容量が設定値の 80% で動作しているときに、Integration Server がより円滑にドキュメント処理を実行していると判断したとします。この場合、Integration Server Administrator を使用して、それらのトリガーの設定済み容量をリセットすることができます。すなわち、ドキュメント抽出またはドキュメント処理のためのスレッドまたはメモリの使用量に影響するトリガーのプロパティは、Integration Server Administrator を使用して設定することができます。このように設定できるプロパティには、トリガーキューの容量、補充レベルおよびトリガーの実行スレッドの最大数があります。

webMethods messaging triggerの容量および補充レベルを変更する機能は、Broker からドキュメントを受信する webMethods messaging triggerに限定されます。Universal Messaging からドキュメントを受信するトリガーの容量の変更は、設計時に限られ、実行時には変更できません。Universal Messaging からドキュメントを受信する webMethods messaging triggerの容量を設定するには、Designer でトリガーを編集します。なお、Universal Messaging からドキュメントを受信する webMethods messaging triggerには補充レベルがありません。

webMethods messaging triggerのプロパティを変更するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [webMethods トリガー] に移動します。
3. [個別の **webMethods Messaging Trigger** 制御] で、プロパティを編集するトリガーの名前をクリックします。
4. [**プロパティの編集**] をクリックします。
5. [**プロパティ**] で、次のうちの 1 つまたは複数を指定します。

プロパティ	指定内容
[キューの容量]	トリガーキューに格納できるドキュメントの最大数。 **注:**

プロパティ	指定内容
	実行時に Integration Server Administrator からキュー容量を変更する機能は、Broker からドキュメントを受信する webMethods messaging triggerにのみ適用されます。Universal Messaging からドキュメントを受信する webMethods messaging triggerの容量を設定するには、Designer でトリガーを編集します。
[キューの補充レベル]	Integration Server が Broker からキューにさらにドキュメントを抽出するまでの、トリガーキューに残っている未処理ドキュメントの数。**[キューの補充レベル]** の値は **[キューの容量]** 以下である必要があります。 **注:** **[キューの補充レベル]** は、Broker からドキュメントを受信する webMethods messaging triggerにのみ適用されます。Universal Messaging からドキュメントを受信する webMethods messaging triggerには、補充レベルは適用されません。Universal Messaging からドキュメントを受信するトリガーについては、**[キューの補充レベル]** フィールドに [該当せず] と表示されます。
[実行スレッドの最大数]	Integration Server が並行して処理できるドキュメントの最大数。同時実行トリガーに対してのみ実行スレッドの最大数の値を指定することができます。逐次実行トリガーの場合、このフィールドには [None] と表示されます。

トリガーのキュー容量、補充レベルおよび実行スレッドの最大数の詳細とガイドラインについては、『*webMethods Service Development Help*』を参照してください。

6. このトリガーのプロパティの変更をクラスタのすべてのサーバに適用するには、**[クラスタ全体に変更内容を適用]** チェックボックスをオンにします。

   このチェックボックスが表示されるのは、適切に設定されたクラスタに属する現在の Integration Server が、トリガーの変更をクラスタ全体で同期するように設定されている場合のみです。トリガー管理の変更をクラスタ全体で同期するように Integration Server を設定する場合の詳細については、900 ページの「webMethods Messaging Trigger管理のクラスタ同期」を参照してください。

7. **[変更内容の保存]** をクリックします。

注:
watt.server.trigger.managementUI.excludeList プロパティを使用することによって、表示されるトリガーのリストをフィルタすることができます。このプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

トリガーサービスの再試行およびシャットダウン要求の管理

Integration Server がトリガーサービスを再試行している間は、トリガーサービスの実行が成功するか最大再試行回数に達するまで、Integration Server はシャットダウン要求を無視します。

このように無視することによって Integration Server はシャットダウンする前にドキュメント処理を完了できます。

ただし、トリガーサービスのすべての再試行を完了せずに Integration Server をシャットダウンする設定が必要な場合もあります。Integration Server には Integration Server のシャットダウン要求に応じてトリガーサービスの再試行プロセスを中断するためのサーバパラメータ watt.server.trigger.interruptRetryOnShutdown があります。

次の表は、`watt.server.trigger.interruptRetryOnShutdown` パラメータに指定可能な値および関連する動作を示しています。

値	説明
`false`	Integration Server のシャットダウン要求が発生してもトリガーサービスの再試行プロセスを中断しません。Integration Server は、最大再試行回数に達するかまたはトリガーサービスの実行が成功した後にのみシャットダウンします。これがデフォルト値です。 **重要:** watt.server.trigger.interruptRetryOnShutdown を「false」に設定している場合にトリガーが成功するまで再試行するように設定すると、トリガーサービスは無限再実行の状況に陥る可能性があります。再試行の原因となっている一時的なエラー状態が解決されない限り、Integration Server は指定した再試行間隔でサービスを再実行し続けます。トリガーサービスの実行中はトリガーの無効化やサーバのシャットダウンができないため、トリガーサービスが無限再試行状態に陥ると、Integration Server がシャットダウン要求に応答しなくなる可能性があります。無限再試行状態から抜け出すには、watt.server.trigger.interruptRetryOnShutdown を「true」に設定します。変更は直ちに有効になります。
`true`	Integration Server のシャットダウン要求に応じてトリガーサービスの再試行プロセスを中断します。具体的には、Integration Server はシャットダウン要求の発生後、現在のサービス再試行が完了するまで待機します。トリガーサービスをもう一度再試行する必要がある (ISRuntimeException が原因でサービスが終了した) 場合でも、Integration Server は再試行プロセスを停止し、シャットダウンします。再起動時、転送手段 (Broker、Universal Messaging、またはローカルパブリッシュの場合は一次ストア) は、処理のためにドキュメントをトリガーに再度デリバーします。 **注:** トリガーサービスの再試行プロセスが中断され、ドキュメントがトリガーに再デリバーされると、転送手段によってドキュメントの再デリバー回数が増やされます。トリガーの重複検出の設定で、ドキュメント履歴データベースやドキュメントリゾルバサービスを使用せずに重複検出を実行するように設定している場合、Integration Server では再デリバードキュメントを「インダウト」と見なすためドキュメント処理は実行されません。

注:
watt.server.trigger.interruptRetryOnShutdown パラメータの値を変更すると、変更は直ちに有効になります。

webMethods Messaging Triggerのポーリング要求の遅延

webMethods messaging triggerの非アクティブ時にポーリング要求を遅延させて、ドキュメントの抽出に使用される Integration Server および Broker のリソースを節約することができます。ポーリング遅延を使用すると、Integration Server はポーリング遅延時間を待機した後で、Broker からメッセージを要求します。トリガーの非アクティブ状態が継続する場合に、ポーリング遅延が徐々に増加するように設定できます。

注:
webMethods messaging triggerは、前回のメッセージングプロバイダへの要求でドキュメントを受信しなかった場合に、非アクティブと見なされます。

ポーリング遅延の間は、webMethods messaging triggerは新しいドキュメントを抽出しません。ただし、遅延中は webMethods messaging triggerはメッセージングプロバイダリソースを使用せず、また webMethods messaging triggerはメッセージングプロバイダをポーリングするための Integration Server リソースも使用しません。

ポーリング遅延を使用するには、以下の項目を指定する必要があります。

- トリガーが webMethods messaging triggerに対してメッセージをポーリングするまで経過する必要がある遅延時間。watt.server.control.triggerInputControl.delays パラメータによって、遅延の長さが決まります。徐々に増加する遅延を使用する場合、整数のカンマ区切りリストを使用して、増加する遅延を指定します。watt.server.control.triggerInputControl.delays パラメータはミリ秒単位で指定し、デフォルト値は「500, 1000, 1500, 5000」です。
- Integration Server が非アクティブの webMethods messaging triggerに対してポーリング遅延を使用する間隔。トリガーが非アクティブのままである場合、この間隔により Integration Server がポーリング要求間の遅延を増加させるタイミングが決まります。この間隔は watt.server.control.triggerInputControl.delayIncrementInterval パラメータによって定義します。このパラメータはミリ秒で指定し、デフォルト値は 1000 です。

Integration Server では、次の式を使用して、webMethods messaging triggerの Broker からのドキュメント抽出のポーリング遅延を決定します。

非アクティブ時間 / 間隔 = 遅延

ここで、

- 非アクティブ時間は、webMethods messaging triggerの非アクティブ継続時間です (ミリ秒)。つまり、非アクティブ時間は、空のトリガークライアントキューを Integration Server が最後に Broker にポーリングしてからの経過時間で (ミリ秒)、Broker にはトリガーに対するメッセージがない状態です。
- 間隔は、watt.server.control.triggerInputControl.delayIncrementInterval の値です。
- 遅延は、除算の結果を最も近い整数に切り捨てた数値で、watt.server.control.triggerInputControl.delays で指定される遅延時間のカンマ区切りリスト内のインデックス位置に対応します。

注:
watt.server.control.triggerInputControl.delays または watt.server.control.triggerInputControl.delayIncrementInterval で指定された値を解析するときに例外が発生した場合、Integration Server は両方の設定パラメータをデフォルト値にリセットします。

Integration Server が webMethods Messaging Triggerのポーリング要求を遅延させる仕組み

ここでは、webMethods messaging trigger がアクティブではない場合に Integration Server がドキュメントに対して Broker をポーリングする方法について、ステップごとに例に示します。この例は、関連するサーバ設定パラメータのデフォルト値に基づいています。具体的なパラメータ値は以下のとおりです。

- watt.server.control.triggerInputControl.delayIncrementInterval = 10000
- watt.server.control.triggerInputControl.delays = 500,1000,1500,5000

次の表では、Integration Server が、アクティブではない webMethods messaging trigger のドキュメントに対して Broker をポーリングするときの各ステップについて説明します。

手順	説明
1	webMethods messaging triggerが非アクティブになります。 Broker への前回の要求でトリガーがドキュメントを受信しなかった場合、Integration Server は webMethods messaging triggerが非アクティブであると見なします。
2	Integration Server は、式「非アクティブ時間 / 間隔 = 遅延」を使用してポーリング遅延の長さを算定し、ポーリング要求の遅延を開始します。上記のとおりサーバパラメータのデフォルト値を使用して、以下の操作を行います。 Integration Server は、次の式を使用してポーリング遅延の長さを算定します。 0 / 1000 = 0 watt.server.control.triggerInputControl.delays プロパティでインデックス位置 0 の遅延は 500 です。Integration Server は 500 ミリ秒待機してから Broker をポーリングしてドキュメントを確認します。
3	webMethods messaging triggerに対するポーリング要求が何もドキュメントを返さなかった場合、Integration Server は、次の式を使用してポーリング遅延の長さを算定します。 500 / 10000 = 0.05 Integration Server は、算出された遅延時間の 0.05 を 0 に切り捨てます。watt.server.control.triggerInputControl.delays プロパティでインデックス位

手順	説明
	置 0 の遅延は 500 です。Integration Server は 500 ミリ秒待機してから Broker をポーリングしてドキュメントを確認します。
4	トリガーの非アクティブ状態が継続している場合、Integration Server は、式「非アクティブ時間 / 間隔 = 遅延」を使用してポーリング遅延を引き続き算定します。非アクティブ時間は、毎回 500 ミリ秒ずつ増加します。
5	トリガーの非アクティブ時間が 10000 ミリ秒に達すると、Integration Server は次の式を使用してポーリング遅延の長さを算定します。 10000 / 10000 = 1 watt.server.control.triggerInputControl.delays プロパティでインデックス位置 1 の遅延は 1000 です。Integration Server は 1000 ミリ秒待機してから Broker をポーリングしてドキュメントを確認します。
6	トリガーの非アクティブ状態が継続している場合、Integration Server は、式「非アクティブ時間 / 間隔 = 遅延」を使用してポーリング遅延を引き続き算定します。非アクティブ時間は、毎回 1000 ミリ秒ずつ増加します。 **注:** Integration Server は、算出された遅延時間を最も近い整数に切り捨てます。たとえば、トリガーが 15000 ミリ秒の間非アクティブであった場合、ポーリング遅延の長さは式「15000 / 10000 = 1.5」で算出されます。Integration Server は、この数値を 1 に切り捨てて、watt.server.control.triggerInputControl.delays property のインデックス位置 2 の値である 1000 ミリ秒をポーリング遅延として使用します。
7	トリガーの非アクティブ時間が 20000 ミリ秒に達すると、Integration Server は次の式を使用してポーリング遅延の長さを算定します。 20000 / 10000 = 2 watt.server.control.triggerInputControl.delays プロパティでインデックス位置 2 の遅延は 1500 です。Integration Server は 1500 ミリ秒待機してから Broker をポーリングしてドキュメントを確認します。
8	トリガーの非アクティブ状態が継続している場合、Integration Server は、式「非アクティブ時間 / 間隔 = 遅延」を使用してポーリング遅延を引き続き算定します。非アクティブ時間は、毎回 1500 ミリ秒ずつ増加します。
9	トリガーの非アクティブ時間が 30000 ミリ秒に達すると、Integration Server は次の式を使用してポーリング遅延の長さを算定します。 30000 / 10000 = 3 watt.server.control.triggerInputControl.delays プロパティでインデックス位置 3 の遅延は 5000 です。Integration Server は 5000 ミリ秒待機してから Broker をポーリングしてドキュメントを確認します。

手順	説明
10	トリガーの非アクティブ状態が継続している場合、Integration Server は、式「非アクティブ時間 / 間隔 = 遅延」を使用してポーリング遅延を引き続き算定します。非アクティブ時間は、毎回 5000 ミリ秒ずつ増加します。 5000 は watt.server.control.triggerInputControl.delays の配列で指定された最大値であるため、ポーリング遅延は 5000 ミリ秒を超えることはありません。Integration Server は、トリガーがアクティブになるまでこのポーリング遅延を使用します。 **注:** watt.server.control.triggerInputControl.delays の最後の値は、Broker が Broker 上のクライアントキューにドキュメントを配置してから Integration Server がドキュメントを抽出するまでの遅延の最大値となります。
11	Broker へのポーリング要求が 1 つ以上のドキュメントを返した場合、Integration Server はトリガーがアクティブであると見なし、ポーリング遅延の使用を停止します。

Integration Server の以前のバージョンから 10.3 以降への逐次トリガーの移行

Integration Server バージョン 10.3 では Universal Messaging をメッセージングプロバイダとして使用するときは、逐次処理の webMethods messaging trigger は Universal Messaging の逐次継続的サブスクリプションに対応します。Integration Server バージョン 9.9 から10.2 では、逐次処理の webMethods messaging trigger は Universal Messaging の優先継続的サブスクリプションに対応していました。Integration Server 9.9 以前のバージョンでは、逐次処理の webMethods messaging trigger は Universal Messaging の共有継続的サブスクリプションに対応していました。

Integration Server 10.3 以降で作成されたすべての webMethods messaging trigger は、逐次継続的サブスクリプションに対応します。ただし、Integration Server バージョン 9.9 から 10.2 へ移行された逐次トリガーは尚、優先名オブジェクト (継続的サブスクリプション) に対応します。Integration Server 9.9 以前のバージョンから移行された逐次トリガーは尚、共有名オブジェクト (継続的サブスクリプション) に対応します。その結果、逐次トリガーと継続的サブスクリプションの同期が外れます。移行された逐次トリガーおよび継続的サブスクリプションを同期させるには、以下のいずれかを実行する必要があります。

- Universal Messaging 10.3 以降を新たにインストールする場合 (つまり Universal Messaging サーバが移行されていない場合) は、Integration Server を開始する際にパブリッシュ可能なドキュメントタイプを、Designer またはビルトインサービス pub.publish:syncToProvider を使用してプロバイダと同期させます。パブリッシュ可能なドキュメントタイプの同期により Integration Server は webMethods messaging trigger を再ロードします。Integration Server は各逐次トリガーに対して逐次継続的サブスクリプションを作成します。
- 旧バーションから移行した Universal Messaging 9.9 以降のインストール版を使用している場合は、逐次トリガーに対応する継続的サブスクリプションを削除して再作成する必要があります。トリガーに関連付けられた継続的サブスクリプションの削除と再作成についての詳

細は、910 ページの「webMethods Messaging Trigger と Universal Messaging 上の継続的サブスクリプションの同期 」を参照してください。

webMethods Messaging Trigger と Universal Messaging 上の継続的サブスクリプションの同期

webMethods messaging triggerと Universal Messaging 上の関連付けられた継続的サブスクリプションは非同期になることがあります。たとえば、クライアントプリフィックスを共有する Universal Messaging 接続エイリアスを使用している webMethods messaging trigger の処理モードを変更しても、Integration Server は Universal Messaging 上のトリガーに対応する継続的サブスクリプションの削除、再作成を行いません。よって、Integration Server 上のトリガーは Universal Messaging 上の継続的サブスクリプションと非同期になります。webMethods messaging trigger と関連付けられた継続的サブスクリプションを同期するには、継続的サブスクリプションを削除して再作成する必要があります。

注:
逐次処理モードの webMethods messaging trigger は Universal Messaging での優先継続的サブスクリプションに対応します。逐次処理モードの webMethods messaging trigger は Universal Messaging での共有継続的サブスクリプションに対応します。Integration Server 10.3 よりも前のバージョンで作成された逐次トリガーが対応する Universal Messaging での継続的サブスクリプションのタイプの詳細については、次の項目を参照してください。 909 ページの「Integration Server の以前のバージョンから 10.3 以降への逐次トリガーの移行」

webMethods messaging trigger と Universal Messaging 上の継続的サブスクリプションを同期するには

- 次のいずれかの手順に従います。
  - webMethods messaging trigger が Universal Messaging に接続されている Integration Server にのみあり、Universal Messaging 接続エイリアスの [**クライアントプリフィックスの共有**] が [**いいえ**] に設定されている場合、トリガーを開始して対応する継続的サブスクリプションを削除し、再作成します。トリガーが使用している Universal Messaging 接続エイリアスをいったん無効にしてから有効にするとトリガーが開始されます。

    注:
    サーバが再起動すると Integration Server がトリガーを開始します。
  - 複数の Integration Server が Universal Messaging に接続されている場合や、Universal Messaging 接続エイリアスの [**クライアントプリフィックスの共有**] プロパティが [**はい**] に設定されている場合は、Universal Messaging Enterprise Manager を使用して継続的サブスクリプションを削除する必要があります。継続的サブスクリプションがすべて排出され新しいドキュメントが送信されてこないときに、継続的サブスクリプションを削除するように注意してください。継続的サブスクリプションを削除する前に、ドキュメントパブリッシャーを休止する必要があるかもしれません。トリガーが使用している Universal Messaging 接続エイリアスを一度無効にしてから有効にすることにより、トリガーの継続的サブスクリプションを作成します。

47 JMS トリガーの管理

- JMS トリガーの管理の概要 912
- JMS トリガーの検索 912
- JMS トリガーのステータスと状態について 913
- JMS トリガーで使用するスレッドの数の制御 918
- Integration Server セッションの再利用の設定 921
- JMS セッションの再利用の設定 921
- 同時実行 JMS トリガーのポーリング要求の遅延 921
- JMS トリガーの開始が失敗したときに実行される処理 925
- WS エンドポイントトリガーについて 927

JMS トリガーの管理の概要

Integration Server および Integration Server Administrator には、JMS トリガーおよび JMS トリガーによって使用されるリソースを管理するための手段が用意されています。具体的には、Integration Server Administrator に備わっている制御を使用して次のようなことができます。

- JMS トリガーに使用されるサーバスレッドの最大数を増加または減少させる。
- 同時実行 JMS トリガーの実行スレッドの最大数を変更する。
- 1 つまたは複数の JMS トリガーを有効化、無効化または一時停止する。
- JMS トリガーがポーリングしてメッセージを確認する頻度を少なくする。

[**JMS トリガー**] ページには、JMS トリガーの管理の大部分の機能が含まれています。[**メッセージング**] > [JMS トリガー] でアクセスできるこのページには、Integration Server にあるすべての JMS トリガー、および各トリガーの概要が表示されます。概要には、トリガーの現在のステータス、状態、およびスレッドの使用状況が含まれます。また、トリガーによって使用される JMS 接続エイリアス、トリガーがサブスクライブする宛先、トリガーの処理モードなどの設定情報も含まれます。Integration Server Administrator は、表示されるトリガーのリストのフィルタまたは特定のトリガーの検索に使用できる検索機能を備えています。

注:
Integration Server Administrator の [**JMS トリガー**] ページには、標準 JMS トリガーのテーブルおよび SOAP-JMS トリガーのテーブルが表示されます。ただし、Integration Server Administrator に SOAP-JMS トリガーのテーブルが表示されるのは、Integration Server のパッケージに SOAP-JMS トリガーが含まれており、そのパッケージがロードされている場合のみです。

ここでは、JMS トリガーの管理の詳細について説明します。

注:
特に指定のない限り、「JMS トリガー」という用語には SOAP-JMS トリガーも含まれます。

JMS トリガーの検索

Integration Server にある JMS トリガーの数が増加すると、Integration Server Administrator の [**メッセージング**] > [JMS トリガー] ページで特定の JMS トリガーを探しにくくなることがあります。[**MS トリガー**] ページでの JMS トリガーの検索を助けるため、または表示されるトリガーのリストをフィルタするため、Integration Server Administrator には JMS トリガーを検索する機能が用意されています。JMS トリガーを検索するときは、1 つ以上の次の条件を指定できます。

- JMS トリガー名
- JMS トリガーで使用される JMS 接続エイリアス名
- JMS トリガーがサブスクライブする宛先

Integration Server Administrator の [**メッセージング**] > [JMS トリガー] ページで JMS トリガーを検索するときは、次の情報に注意してください。

- 検索条件は AND 検索です。たとえば、JMS トリガー名と JMS 接続エイリアスを指定した場合、両方の条件を満たす JMS トリガーが検索されます。
- ワイルドカード文字のアスタリスク (*) を使用できます。検索条件として使用される文字列の前後にワイルドカードを指定できます。Integration Server では、JMS トリガーの検索で他のワイルドカード文字の使用をサポートしていません。
- 検索では大文字と小文字が区別されます。

JMS トリガーを検索するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [JMS トリガー] に移動します。
3. [**トリガーの検索**] をクリックします。

   検索条件フィールドが表示されます。
4. 次のうちの 1 つまたは複数に検索条件を指定します。

条件	指定内容
[トリガー名]	完全修飾 JMS トリガー名またはその一部。トリガー名の一部を指定する場合、1 つ以上のワイルドカード文字を含める必要があります。
[接続エイリアス名]	JMS トリガーが JMS プロバイダからメッセージを抽出するときに使用する JMS 接続エイリアス名。
[宛先]	JMS トリガーがサブスクライブする宛先。

5. [**検索**] をクリックします。

   指定した条件を満たす JMS トリガーが検索され、[**JMS トリガー**] ページに検索結果が表示されます。検索条件をさらに入力し、表示された JMS トリガーのリストのフィルタリングを続けることができます。検索結果を消去して、Integration Server のすべての JMS トリガーを表示するには、[**すべてのトリガーの表示**] をクリックします。

JMS トリガーのステータスと状態について

Integration Server Administrator の [JMS トリガー] ページには、JMS トリガーのステータスと状態に関する情報が表示されます。JMS トリガーの状態は、トリガーが有効、無効、一時停止中のいずれであるかを示します。ステータスはトリガーが実行されているかどうかを示します。

JMS トリガーは次のいずれかの状態にできます。

トリガーの状態	説明
有効	JMS トリガーを使用できます。JMS トリガーがメッセージを受信し処理するには、JMS トリガーを有効にする必要があります。 有効なトリガーのステータスが [実行されていない] になることがあります。このステータスは、メッセージを受信、処理しないことを意味します。有効な JMS トリガーは、無効な JMS 接続エイリアス、トリガーによってスローされた例外、起動時のトリガーのエラーなどの理由で、無効になることがあります。
[無効]	JMS トリガーは使用できません。Integration Server は、JMS トリガーに対してメッセージの抽出も処理も行いません。JMS トリガーは、有効化されるまでこの状態のままです。
[一時停止中]	JMS トリガーは実行中で、JMS プロバイダに接続されています。Integration Server はメッセージの抽出を停止しましたが、既に抽出済みのメッセージの処理は続行します。Integration Server は、サーバの再起動時または JMS トリガーを含むパッケージの再ロード時に JMS トリガーを自動的に有効化します。

リストされた JMS トリガーがトリガーグループヘッダーである場合、[**状態**] 列にはトリガーグループのすべての JMS トリガーの状態の概要が表示されます。

- トリガーグループのすべてのトリガーが同じ状態である場合、トリガーグループヘッダーにはその状態が表示されます。
- トリガーグループの少なくとも 1 つのトリガーが有効な場合、トリガーグループヘッダーには [有効 (*n* の内 *x*)] と表示されます。ここで *x* は有効なトリガーの数です。*n* はトリガーグループのトリガー総数です。
- 有効なトリガーがなくグループの少なくとも 1 つのトリガーが一時停止中である場合、トリガーグループヘッダーには [一時停止中 (*n* の内 *x*)] と表示されます。ここで *x* は一時停止中のトリガーの数です。*n* はトリガーグループのトリガー総数です。

JMS トリガーのステータスは [実行中] または [実行されていない (*reason*)] を表示することができます。*reason* は、[実行されていない (トリガー無効)] など、トリガーが実行されていない理由を識別します。

リストされた JMS トリガーがトリガーグループヘッダーである場合、[**ステータス**] 列にはトリガーグループのすべての JMS トリガーのステータスの概要が表示されます。

- グループのすべてのトリガーが実行されている場合、[**ステータス**] 列には [実行中] が表示されます。
- グループの少なくとも 1 つのトリガーが実行されている場合、[**ステータス**] 列には [実行中 (*n* の内 *x*)] が表示されます。
- グループのどのトリガーも実行されていない場合、[**ステータス**] 列には [実行されていない] が表示されます。

JMS トリガーの有効化、無効化および一時停止

JMS トリガーの状態を変更することによって、JMS トリガーおよび JMS トリガーが消費するリソースの量を管理できます。

Integration Server Administrator では次のようなことができます。

- すべての JMS トリガーの有効化、無効化または一時停止
- 特定のタイプのすべての JMS トリガーの有効化、無効化または一時停止 (標準または SOAP-JMS)
- 特定の JMS トリガーの有効化、無効化または一時停止
- JMS トリガーグループのすべてのトリガーの有効化、無効化または一時停止

サーバリソースを解放する迅速な方法として、すべての JMS トリガーの状態を一度に変更することができます。この方法は、Integration Server が高負荷で動作していて、追加のリソースをすぐに必要とする場合に特に有効です。

特定の JMS トリガーの状態を変更するのは、次のような場合です。

- バックエンドシステムの保守が必要な場合や応答が鈍ってきている場合は、バックエンドシステムとやりとりしている JMS トリガーを一時停止します。バックエンドシステムが使用可能になったら、JMS トリガーを有効化します。
- すべての JMS トリガーを一時停止または無効化した後で、特定の JMS トリガーを有効化できます。たとえば、サーバが通常にはない重い負荷で動作している場合は、まずすべての JMS トリガーを一時停止し、その上で重要なプロセスに関わる JMS トリガーから順に少しずつ JMS トリガーを有効にしていくことができます。
- メッセージの処理中に致命的なエラーが発生したために、Integration Server によって逐次実行 JMS トリガーが自動的に一時停止された場合。致命的なエラーを処理するための JMS トリガーの設定については、『*webMethods Service Development Help*』を参照してください。
- JMS 用にプロバイダ Web サービスエンドポイントエイリアスを作成したために、WS エンドポイントトリガーが作成された場合。

重要:
1 つ以上のプロバイダ Web サービス記述子のリスナーとして動作する SOAP-JMS トリガーを無効化または一時停止すると、Integration Server はこれらの Web サービス記述子のメッセージを抽出しません。

次の手順では、Integration Server Administrator を使用してすべてまたは個別の JMS トリガーの状態を変更する方法を説明します。

JMS トリガーを有効化、無効化または一時停止するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [JMS トリガー] に移動します。

3. すべての JMS トリガーの状態を変更する場合は、以下の手順に従います。

   a. [個別 JMS トリガーコントロール] の [**状態**] 列で、[**一括編集**] をクリックします。

   b. [**トリガーの状態の編集**] の [**新規の状態**] リストで、すべての JMS トリガーに適用させる状態を選択します。

   c. [**JMS トリガータイプ**] リストで、以下のいずれかを選択します。

選択項目	目的
[標準]	状態変更を標準 JMS トリガーにのみ適用します。
[SOAP]	状態変更を SOAP-JMS トリガー (WS エンドポイントトリガーを含む) にのみ適用します。
[すべて]	状態変更をすべての JMS トリガーに適用します。

4. 特定の JMS トリガーの状態を変更する場合は、以下の手順に従います。

   a. [個別 JMS トリガーコントロール] で、有効化、無効化または一時停止する JMS トリガーの行にある [**状態**] 列でテキストをクリックします。

   b. [**メッセージング**] > [JMS トリガー] > [トリガーの状態の編集] ページの [**新規の状態**] リストで、この JMS トリガーに適用する状態を選択します。

5. JMS トリガーグループ内のすべての JMS トリガーの状態を変更する場合は、以下の手順に従います。

   a. [個別 JMS トリガーコントロール] で、すべてのトリガーを有効化、無効化または一時停止する対象トリガーグループの JMS トリガーグループヘッダーの行にある [**状態**] 列でテキストをクリックします。

   b. [**メッセージング**] > [JMS トリガー] > [トリガーの状態の編集 - JMS トリガーグループ] ページの [**新規の状態**] リストで、この JMS トリガーに適用する状態を選択します。

6. [**変更内容の保存**] をクリックします。

メモ:

- JMS トリガーを無効にする場合は、最初に JMS トリガーを一時停止して、処理中のすべてのスレッドが完了するまで待ちます。その後で、JMS トリガーを無効にします。現在 JMS トリガーで使用されているスレッドの数は、[**メッセージング**] > [JMS トリガー] ページで確認できます。
- JMS トリガーを無効にすると、Integration Server によって以下の処理が実行されます。

- JMS トリガーが再試行を待機している場合、Integration Server はその JMS トリガーの処理を中断します。
- JMS トリガーが現在メッセージの処理中である場合、Integration Server は指定された時間待機してから、メッセージの処理を停止するよう JMS トリガーに強制します。指定された時間内に処理が完了しなかった場合、Integration Server は JMS トリガーのメッセージを受信するために使用されているメッセージコンシューマを停止して、JMS セッションを閉じます。この時点で、JMS トリガーのサーバスレッドは完了するまで実行し続けます。ただし、JMS トリガーで、処理中のメッセージの完了を確認することはできません。メッセージの配信モードが永続モードに設定されている場合、完了を確認できないことによってメッセージの重複が発生する可能性があります。

  Integration Server が JMS トリガーを無効にする要求を受け取ってからそのトリガーに停止を強制するまで待機する時間は、`watt.server.jms.trigger.stopRequestTimeout` プロパティで指定します。

- 宛先などの管理対象オブジェクトは Integration Server 外で設定されるため、JMS トリガーを無効にしてもサブスクリプションには何の影響もありません。
- JMS トリガーを一時停止した時点でその JMS トリガーがメッセージを処理中だった場合、JMS トリガーはそれらのメッセージの処理を完了します。JMS トリガーは JMS プロバイダへのメッセージの受信も確認します。
- 組み込みサービスを使用して、1 つまたは複数のトリガーを有効化、無効化および一時停止することができます。pub.trigger:disableJMSTriggers、pub.trigger:enableJMSTriggers および pub.trigger:suspendJMSTriggers サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

コンシューマエラー後に JMS トリガーが自動的に有効化されるように Integration Server を設定する

JMS プロバイダへの接続に問題がある場合、Integration Server は JMS トリガーを無効にします。Integration Server は、JMS トリガーによって使用されたメッセージコンシューマで、JMS プロバイダへの接続に関する問題以外の原因で予期しないエラーが発生した場合も、JMS トリガーを無効にする可能性があります。原因には、次のような問題が考えられます。

- Integration Server と JMS プロバイダ間のネットワークレイテンシ
- Integration Server と JMS プロバイダを分離する広域ネットワーク
- 大きな JMS メッセージの受信
- その他のネットワーク関連の問題

Integration Server を設定して、メッセージコンシューマに JMS プロバイダへの接続に関する問題以外の原因でエラーが発生したときに無効化された JMS トリガーの再有効化を自動的に試行できます。この再有効化を行うには、`watt.server.jms.trigger.retryOnConsumerError` を「true」に設定します。このプロパティを「false」に設定すると、Integration Server は JMS トリガーの無効状態を維持します。デフォルトは「true」です。

JMS トリガーで使用するスレッドの数の制御

Integration Server Administrator を使用して、JMS トリガーで使用できるサーバスレッドの数を制限することができます。デフォルトでは、Integration Server は JMS トリガーに対して最大で 100% のサーバスレッドプールを消費します。ただし、その他の機能の実行にもサーバリソースが必要です。

JMS トリガーのスレッドの使用数の表示

Integration Server Administrator によって、Integration Server 上の JMS トリガーごとに現在使用されているサーバスレッドの数が表示されます。

JMS トリガーのスレッドの使用数を表示するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [JMS トリガー] に移動します。

   Integration Server Administrator によって [**個別標準 JMS トリガー制御**] の [**現在のスレッド**] 列に、現在メッセージの受信および処理に使用されているサーバスレッドの数が JMS トリガー別に表示されます。Integration Server が JMS プロバイダに接続されていない場合は、[**現在のスレッド**] 列に「未接続」と表示されます。

JMS トリガーのスレッドの使用数の調整

Integration Server には、JMS トリガーによって使用されるサーバスレッドの数を調整するために使用できるコントロールが備わっています。これらのコントロールを使用して次のようなことができます。

- すべての JMS トリガーの受信および処理に Integration Server で使用可能なサーバスレッドプールのパーセンテージを設定する。
- 同時実行 JMS トリガー全体で、同じパーセンテージで実行スレッドの最大数を減少させる。この場合、並行 JMS トリガーでメッセージを処理する速度も低下します。

JMS トリガーのスレッドの使用数を調整するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [JMS トリガー] に移動します。
3. [**JMS グローバルトリガー制御の編集**] をクリックします。
4. [**スレッドプールのスロットル**] フィールドに、JMS トリガーで使用できるサーバスレッドプールの最大値に対するパーセンテージを入力します。サーバスレッドプールには、JMS プ

ロバイダからのメッセージの抽出に使用されるスレッドと、メッセージの処理に使用されるスレッドが含まれます。0 より大きい値を入力してください。

5. [**個別トリガー処理のスロットル**] フィールドで、実行スレッドの最大数の設定値に対してサーバが動作するパーセンテージとして、値を選択します。Integration Server によって、このパーセンテージは、すべての同時実行 JMS トリガーの実行スレッドの最大数の値に適用されます。

   この値は同時実行 JMS トリガーのみに適用されます。

6. [**変更内容の保存**] をクリックします。

メモ:

- [**スレッドプールのスロットル**] に指定したパーセンテージがサーバスレッドプールのサイズに適用されたときに整数にならなかった場合、Integration Server によって端数の切り上げまたは切り捨てが行われ、最も近い整数になります。
- 逐次実行 JMS トリガーは、常に一度に 1 つのメッセージを処理します。逐次実行トリガーの場合、Integration Server はメッセージの受信および処理に同じスレッドを使用します。[**個別トリガー処理のスロットル**] は逐次実行 JMS トリガーには影響しません。
- 同時実行 JMS トリガーはコンシューマのプールを使用してメッセージを受信および処理します。各コンシューマはサーバスレッドプールのスレッドを使用して、メッセージを受信および処理します。同時実行 JMS トリガーは、接続ごとに 1 つの追加スレッドをコンシューマのプール管理専用にします。たとえば、1 つの JMS プロバイダ接続を使用するように設定され、さらに一度に最大 10 個のメッセージを処理するように設定されている同時実行 JMS トリガーでは、最大で 11 個のサーバスレッドを使用できます。また、2 つの JMS プロバイダ接続を使用するように設定され、さらに一度に最大 10 個のメッセージを処理するように設定されている同時実行 JMS トリガーでは、最大で 12 個のスレッドを使用できます。同時実行 JMS トリガーに複数の接続を使用する方法の詳細については、319 ページの「JMS 接続エイリアスの複数接続の許可」を参照してください。
- 同時実行 JMS トリガーが複数の webMethods Broker 接続を使用するように設定されている場合、トリガーの実行スレッドの最大数は、トリガーの [**接続数**] プロパティで指定した接続数よりも小さくはできません。このことは、[**個別トリガー処理のスロットル**] の値を減らす場合でも当てはまります。
- 指定したパーセンテージで減らした結果、同時実行 JMS トリガーの実行スレッド数が整数にならない場合は、Integration Server によって端数の切り上げまたは切り捨てが行われ、最も近い整数に調整されます。ただし、切り捨ての結果、値が 0 に設定される場合は、Integration Server によって 1 に切り上げられます。たとえば、[**個別トリガー処理のスロットル**] を最大値の 10% に減少させた場合、実行スレッドの最大数の値が 12 に設定されている同時実行 JMS トリガーの値は 1 に調整されます (Integration Server によって 1.2 が 1 に切り下げされます)。実行スレッドの最大数の値が 4 に設定されている同時実行 JMS トリガーの値は 1 に調整されます (Integration Server によって 0.4 が 1 に切り上げられます)。
- [**個別トリガー処理のスロットル**] の値を減らして変更を保存しても、Integration Server では、現在同時実行 JMS トリガーを実行しているスレッドを終了して、調整された最大値に合わせるということは行いません。Integration Server では、同時実行 JMS トリガーのメッ

セージを処理しているサーバスレッドは、処理を完了するまで実行できます。Integration Server は、同時実行 JMS トリガーの処理を実行しているスレッドの数が、調整された最大値を下回るまで待機してから、その JMS トリガー用に次のサーバスレッドをディスパッチします。

同時実行 JMS トリガーで複数のスレッドを使用するためのしきい値の設定

デフォルトでは、同時実行 JMS トリガーは処理するメッセージが複数ある場合に、複数のサーバスレッドの使用を開始します。Integration Server は、JMS トリガーがメッセージの抽出と処理に使用できるスレッドの最大数を使用するまで、トリガーが使用するサーバスレッドの数を増やします。JMS トリガーが使用できるスレッドの最大数は、その JMS トリガーに設定された [**実行スレッドの最大数**] プロパティ値によって決まります。

同時実行 JMS トリガーが使用するスレッドの数を制限するには、JMS トリガーがマルチスレッド処理になるタイミングを指定するしきい値を設定します。1 日の特定の時刻にメッセージが殺到し、同時実行 JMS トリガーに追加の処理リソースが必要になるものの、1 日のうちの大半は JMS トリガーの受信するメッセージの数が比較的少ない場合に、しきい値が有効です。メッセージの受信速度が低い場合、JMS トリガーは単一のスレッドのみでメッセージを効率的に処理できます。

このしきい値は、JMS トリガーが JMS プロバイダに対して行った、メッセージ抽出に連続して成功した要求数に基づいています。しきい値を使用している場合でも、同時実行 JMS トリガーは、初めのうちは単一のスレッドを使用してメッセージを抽出し、処理します。メッセージ抽出に連続して成功した要求数が指定されたしきい値に達すると、JMS トリガーはマルチスレッド処理になります。Integration Server が JMS トリガーに使用できるスレッドの最大数は、[**実行スレッドの最大数**] プロパティの値に 1 を加えた値となります(同時実行 JMS トリガーがマルチスレッド処理の場合、Integration Server はサーバスレッドをトリガーのメッセージコンシューマプールの管理専用とします)。同時実行 JMS トリガーは、スレッドによって作成されたメッセージコンシューマがタイムアウトするまで、マルチスレッド処理の状態を維持します。メッセージコンシューマがタイムアウトするタイミングは、watt.server.jms.trigger.pooledConsumer.timeout プロパティ値で指定します。コンシューマがタイムアウトすると、JMS トリガーは関連するサーバスレッドを解放します。最終的に、受信するメッセージがなくなると、同時実行 JMS トリガーは単一のスレッドを使用する状態に戻ります。

しきい値を使用して、JMS トリガーがシングルスレッド処理からマルチスレッド処理に移行するタイミングを指定するには、watt.server.jms.trigger.concurrent.consecutiveMessageThreshold サーバプロパティを 0 (ゼロ) より大きい正の整数に設定します。このプロパティは、同時実行 JMS トリガーがマルチスレッド処理されるようになるまでに、JMS プロバイダからのメッセージ抽出に連続して成功する必要のある要求数を示しています。たとえば、このプロパティを 1000 に設定した場合、JMS プロバイダからのメッセージ抽出に連続して成功した要求数が 1000 になると、同時実行 JMS トリガーはマルチスレッド処理になります。このプロパティのデフォルトは 0 で、これは同時実行 JMS トリガーをマルチスレッド処理にするしきい値の使用が無効であることを示します。しきい値の使用が無効の場合、処理するメッセージが複数あれば同時実行 JMS トリガーは常にマルチスレッド処理されます。

注:

同時実行 JMS トリガーがマルチスレッド処理になるタイミングを指定するしきい値は、Integration Server のすべての同時実行 JMS トリガーに適用されます。これには、標準 JMS トリガー、SOAP-JMS トリガーおよび WS-エンドポイントトリガーが含まれます。JMS トリガーごとにしきい値を設定することはできません。

Integration Server セッションの再利用の設定

JMS トリガーがサービスを呼び出すたびに同じ Integration Server セッションを再利用すると、システムパフォーマンスを向上させることができます。Integration Server セッションを再利用すると、サービスの呼び出しごとに Integration Server で必要となるオーバーヘッドの量が減少します。Integration Server は、トランザクションに使用される JMS トリガーとトランザクションに使用されていない JMS トリガーおよび SOAP-JMS トリガーでの Integration Server セッションの再利用をサポートしています。

`watt.server.jms.trigger.reuseSession` サーバ設定パラメータを「true」に設定すると、サービスの呼び出しごとに同じ Integration Server セッションを再利用するよう JMS トリガーを設定できます。デフォルトでは、このパラメータは「false」に設定されます。watt.server.jms.trigger.reuseSession の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

再利用されたセッションでサービスを呼び出すと、それぞれが同じセッション ID を使用します。一意のセッション ID を必要とするサービスではセッションを再利用できません。

注:
Adapter for SAP を使用する場合、同時 JMS トリガーに pub.sap.client:lockSession および pub.sap.client:releaseSession サービスを呼び出すマルチステップ SAP トランザクションを実行するトリガーサービスがあるときは、`watt.server.jms.trigger.reuseSession` を「false」に設定します。これらのサービスには一意のセッションが必要です。

JMS セッションの再利用の設定

複数のトランザクションで JMS セッションを再利用すると、JMS プロバイダのオーバーヘッドを削減できます。トランザクションに使用されていない JMS トリガーおよび SOAP-JMS トリガーは、メッセージを受信して処理した後、JMS セッションを自動的に再利用します。トランザクションに使用される JMS トリガーおよび SOAP-JMS トリガーでこれと同じ動作を設定するには、`watt.server.jms.trigger.reuseJmsTxSession` サーバ設定パラメータを「true」に設定します。使用している JMS プロバイダで、トランザクションに使用される JMS セッションを再利用できない場合は、このパラメータを「false」に設定します。デフォルトでは、このパラメータは「true」に設定されています。`watt.server.jms.trigger.reuseJmsTxSession` サーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

同時実行 JMS トリガーのポーリング要求の遅延

JMS トリガーの非アクティブ時にポーリング要求を遅延させて、同時実行 JMS トリガーがメッセージポーリングに使用するリソースを削減することができます。ポーリング遅延機能が有効である場合、非アクティブな同時実行 JMS トリガーは指定されたポーリング遅延時間を待機してから、JMS プロバイダをポーリングしてメッセージを確認します。Integration Server を設定し

て、JMS トリガーが非アクティブである場合に、ポーリング要求間の遅延を徐々に増やすことができます

注:
同時実行 JMS トリガーは、JMS トリガーに関連付けられているどのメッセージコンシューマも前回の JMS プロバイダへの要求でメッセージを受信していない場合、非アクティブと見なされます。

ポーリング遅延の間に、JMS トリガーはいかなるメッセージも受信しません。ただし、JMS トリガーは遅延中に JMS プロバイダのリソースを使用しません。多くのメッセージに対して要求を行うと、現在使用している JMS プロバイダに負荷がかかりすぎる場合、ポーリング遅延を使用することで、非アクティブな同時実行 JMS トリガーによって JMS プロバイダにかかる負荷を軽減できます。

ポーリング要求を遅延させるための Integration Server の設定は、Integration Server のシャットダウンに必要な時間への影響を最小限に抑えます。シャットダウンの一環として、Integration Server は各 JMS トリガーをシャットダウンします。Integration Server は遅延の延長を中断して JMS トリガーのシャットダウンを開始できます。

ポーリング遅延を使用するには、以下の項目を指定する必要があります。

- ポーリング間隔が開始されるまで経過する必要がある遅延の長さ。watt.server.jms.trigger.extendedDelay.delays パラメータによって、遅延の長さが決まります。徐々に増加する遅延を使用する場合、整数のカンマ区切りリストを使用して、増加する遅延を指定します。watt.server.jms.trigger.extendedDelay.delays パラメータはミリ秒単位で指定し、デフォルト値は「0, 1000, 2000, 3000」です。
- Integration Server が非アクティブの同時実行 JMS トリガーに対してポーリング遅延を使用する間隔。JMS トリガーが非アクティブのままである場合、この間隔により Integration Server がポーリング要求間の遅延を増加させるタイミングが決まります。この間隔は watt.server.jms.trigger.extendedDelay.delayIncrementInterval パラメータによって定義します。このパラメータはミリ秒単位で指定し、デフォルト値は 0 です。0 は、Integration Server が非アクティブ同時実行 JMS トリガーに対するポーリング要求間の遅延を使用しないことを意味します。

注:
Universal Messaging からメッセージを受信して、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーは、ここで説明されているメッセージポーリングの遅延動作は使用しません。

JMS トリガーがポーリング要求を遅延させる仕組み

次の表は、Integration Server が非アクティブ同時実行 JMS トリガーに対してポーリング要求を遅延するように設定されている場合に、同時実行 JMS トリガーがどのようにメッセージをポーリングするかについて説明しています。

ステージ	説明
1	同時実行 JMS トリガーが非アクティブになります。

ステージ	説明
	JMS トリガーに関連付けられているどのメッセージコンシューマも前回の JMS プロバイダへの要求でメッセージを受信していない場合、Integration Server は同時実行 JMS トリガーを非アクティブと見なします。
2	watt.server.jms.trigger.extendedDelay.delayIncrementInterval パラメータが 1 以上である場合、ポーリング遅延機能は有効です。JMS トリガーがポーリング要求の遅延を開始します。遅延時間は、watt.server.jms.trigger.extendedDelay.delays パラメーターの最初に指定された整数値です。 たとえば、「watt.server.jms.trigger.extendedDelay.delays = 1000, 2000, 3000」とします。JMS トリガーが非アクティブになると、JMS トリガーは次のように動作します。 1. 1000 ミリ秒待機する 2. JMS プロバイダをポーリングしてメッセージを確認する watt.server.jms.trigger.extendedDelay.delayIncrementInterval パラメータが 0 に設定されている場合、ポーリング遅延機能は無効です。Integration Server は、ポーリング要求間の遅延を使用しません。
3	watt.server.jms.trigger.extendedDelay.delayIncrementInterval で指定された間隔の経過後に、JMS トリガーは watt.server.jms.trigger.extendedDelay.delays の 2 番目に指定された整数値を使用して、ポーリング要求の遅延を開始します。 たとえば、ミリ秒で指定される watt.server.jms.trigger.extendedDelay.delayIncrementInterval が 5000 であるとします。JMS トリガーは、5000 ミリ秒の間、ステージ 2 で説明した頻度で新しいメッセージをポーリングします。5000 ミリ秒が経過すると、JMS トリガーは 2 番目の遅延の延長値の使用を開始します。「watt.server.jms.trigger.extendedDelay.delays = 1000, 2000, 3000」である場合、JMS トリガーは 2000 ミリ秒待機してから JMS プロバイダをポーリングしてメッセージを確認します。
4	watt.server.jms.trigger.extendedDelay.delayIncrementInterval で指定された間隔の経過後に、JMS トリガーは watt.server.jms.trigger.extendedDelay.delays の 3 番目に指定された整数値を使用して、ポーリング要求の遅延を開始します。 たとえば、ミリ秒で指定される watt.server.jms.trigger.extendedDelay.delayIncrementInterval が 5000 であるとします。JMS トリガーは、5000 ミリ秒の間、ステージ 3 で説明した頻度で新しいメッセージをポーリングします。5000 ミリ秒が経過すると、JMS トリガーは 3 番目の遅延の延長値の使用を開始します。「watt.server.jms.trigger.extendedDelay.delays = 1000, 2000, 3000」である場合、JMS トリガーは 3000 ミリ秒待機してから JMS プロバイダをポーリングしてメッセージを確認します。

ステージ	説明
5	watt.server.jms.trigger.extendedDelay.delayIncrementInterval で指定した時間が経過するごとに、JMS トリガーは watt.server.jms.trigger.extendedDelay.delays の次の値を使用して、ポーリング要求間の間隔を変更します。watt.server.jms.trigger.extendedDelay.delays の最後の値に達したら、JMS トリガーは JMS プロバイダからメッセージを抽出するまで、指定されたポーリング遅延を使用します。
6	JMS プロバイダへのポーリング要求が 1 つ以上のメッセージを返した場合、Integration Server は JMS トリガーがアクティブであると見なし、ポーリング遅延の使用を停止します。

延長ポーリング遅延設定の例

以下の例は、同時実行 JMS トリガーの延長ポーリング遅延の設定方法を詳細に示しています。

例 1

延長ポーリング遅延および同時実行 JMS トリガーのポーリング間隔に関するサーバ設定パラメータが次のように設定されていると仮定します。パラメータはミリ秒で指定されていることに留意してください。

- watt.server.jms.trigger.extendedDelay.delays = 100, 200, 1000
- watt.server.jms.trigger.extendedDelay.delayIncrmentInterval = 5000

JMS トリガーが非アクティブになると、JMS トリガーはすぐに 100 ミリ秒のポーリング遅延を待機してから、JMS プロバイダをポーリングしてメッセージを確認します。JMS トリガーは、ポーリング要求間の延長遅延で指定された 100 ミリ秒を待機しながらポーリングを継続します。5000 ミリ秒が経過しても、JMS トリガーが非アクティブのままである場合、JMS トリガーはポーリング要求間に 200 ミリ秒の遅延を使用し始めます。さらに 5000 ミリ秒が経過して JMS トリガーが非アクティブのままである場合、JMS トリガーは 1000 ミリ秒の延長遅延を使用し始めます。JMS トリガーは、ポーリング要求が 1 つ以上のメッセージを返すまで、1000 ミリ秒の延長遅延を使用し続けます。

例 2

同時実行 JMS トリガーに延長ポーリング遅延を使用するように Integration Server が設定されている場合、最初の遅延は JMS トリガーが非アクティブになった直後に発生します。watt.server.jms.trigger.extendedDelay.delayIncrementInterval で指定した間隔の後でのみ、JMS トリガーが延長遅延を使用するようにするには、watt.server.jms.trigger.extendedDelay.delays の最初の値を 0 に設定します。

延長ポーリング遅延および同時実行 JMS トリガーのポーリング間隔に関するサーバ設定パラメータが次のように設定されていると仮定します。パラメータはミリ秒で指定されていることに留意してください。

- watt.server.jms.trigger.extendedDelay.delays = 0, 10000

- watt.server.jms.trigger.extendedDelay.delayIncrmentInterval = 3600000

JMS トリガーは、1 時間 (3600000 ミリ秒) 非アクティブであった後に、10000 ミリ秒のポーリング遅延を待機してから JMS プロバイダをポーリングしてメッセージを確認します。ポーリング要求がメッセージを何も返さない場合、JMS トリガーは 10000 ミリ秒待機してから JMS プロバイダをポーリングしてメッセージを確認します。JMS トリガーは、ポーリング要求がメッセージを返すまで、この方法でポーリングを続行します。

JMS トリガーの開始が失敗したときに実行される処理

JMS 接続エイリアスが開始されると、Integration Server はその JMS 接続エイリアスを使用するすべての JMS トリガーを開始して、JMS プロバイダからメッセージを抽出しようとします。JMS トリガーの開始は、次の理由で失敗することがあります。

- JMS トリガーと JMS プロバイダのどちらか一方または両方が正しく設定されていない (たとえば、トリガーがサブスクライブする宛先の検索名が JNDI プロバイダに存在しない、または Integration Server に宛先からメッセージを抽出するために十分な特権がない)。JMS トリガーまたは JMS プロバイダの設定が正しくないために JMS トリガーが開始されない場合、Integration Server ではそれ以上トリガーを開始するアクションを実行できません。Integration Server では例外がログに記録され、Integration Server Administrator の [JMS トリガー] ページに表示されます。設定が解決され、トリガーが手動で再び開始 (有効化) されるまで、JMS トリガーは非アクティブな状態のままです。

- Integration Server が JMS トリガーを開始しようとする時点で JMS プロバイダが使用できない。JMS プロバイダを使用できない場合、JMS 接続エイリアスも失敗します。接続の失敗のために JMS 接続エイリアスが停止すると、Integration Server は 20 秒ごとに JMS 接続エイリアスの再開始を試みます。JMS プロバイダへの接続が復元されると、JMS 接続エイリアスは再開され、Integration Server では JMS 接続エイリアスを使用するすべての JMS トリガーの開始が試行されます。

- JMS プロバイダで宛先が一時的に使用できない。Universal Messaging の場合、Integration Server で JMS トリガーの開始が試行されると同時にユーザがキュー/チャネルの編集や保守の実行を行うと、キュー/チャネルが一時的に使用できないことがあります。この一時的な状況を処理するため、watt.server.jms.trigger.startupFailure.retryCount server サーバ設定パラメータが 0 より大きい値に設定されており、以下のいずれかに該当する場合、Integration Server ではトリガーの開始を再試行します。

  - JMS トリガーが複数の接続を使用するよう設定されており、少なくとも 1 つのトリガーインスタンスが既に正常に開始されている。JMS トリガーが複数の接続を使用する場合、接続ごとにトリガーの新しいインスタンスが開始されます。

  - javax.jms.InvalidDestinationException が原因で JMS トリガーが失敗した。javax.jms.InvalidDestinationException には、javax.naming.NamingException である JNDI 検索の失敗が含まれないことに注意してください。

  watt.server.jms.trigger.startupFailure.retryCount パラメータ値によって、トリガーの開始が失敗した後に Integration Server が JMS トリガーの開始を再試行する回数の最大値が決まります。トリガーの最初の開始の失敗後、Integration Server は 1000 ミリ秒待機してから、トリガーの開始を再試行します。トリガーの開始が失敗すると、Integration Server では 1000 ミリ秒待機してから、トリガーの開始を再試行します。JMS トリガーが正常に開始されるか、JMS 接続エイリアスの実行が停止するか、または JMS トリガーが無効になるまで、

Integration Server ではこのように処理が継続されます。Integration Server が最大回数の再試行を行った後にトリガーの開始が失敗した場合、Integration Server ではそれ以上トリガーを開始するアクションを実行できません。Integration Server では例外がログに記録され、Integration Server Administrator の [JMS トリガー] ページに表示されます。問題が解決され、トリガーが手動で再び開始 (つまり有効に) されるまで 、JMS トリガーは非アクティブな状態のままです。

watt.server.jms.trigger.startupFailure.retryCount パラメータは 0 以上の整数に設定する必要があります。パラメータを 0 に設定すると (デフォルト)、Integration Server では再試行を行いません。

Integration Server はエイリアスの開始に使用したものと同じスレッドを使用して再試行を実行するため、再試行回数が多すぎると、JMS 接続エイリアスの開始時間が遅れます。

JMS 接続エイリアスが正常に開始した後、正しく設定された JMS トリガーの開始が失敗する可能性があります。これは、JMS 接続エイリアスが Universal Messaging クラスタ内の 1 つの Universal Messaging サーバに接続し、JMS トリガーがクラスタ内の別のサーバに接続する場合に見られることがあります。場合によっては、少し遅れて JMS トリガーが開始することがあります。これらのトリガーが自動的に開始できるようにするため、Integration Server はタスクをスケジュールし JMS トリガーの開始を再試行します。トリガーの再開始タスクの詳細については、926 ページの「JMS トリガーの再開始タスクの仕組み」を参照してください。

JMS トリガーの再開始タスクの仕組み

JMS 接続エイリアスの起動時に JMS トリガーが開始しない場合、Integration Server はJMS トリガーの開始を再試行するためのタスクをスケジュールし、その後 JMS 接続エイリアスの開始に関連するアクティビティを続行します。独自のスレッドで実行するトリガーの再開始タスクは、設定された間隔で、失敗した JMS トリガーの再開始を試みます。デフォルトでは、タスクは 30 秒ごとに実行されます。JMS 接続エイリアスが無効になった、または最大再試行回数に達した場合、JMS 接続エイリアスのすべての失敗したトリガーが正常に開始した後でタスクは完了します。最大再試行回数に達しても JMS トリガーが開始しない場合、残りの JMS トリガーの設定の問題を調査する必要があります。設定の問題が解決されると、Integration Server Administrator を使用してJMS トリガーを手動で再開始することができます。

Integration Server は、エイリアスが正常に起動した際に開始されないトリガーを持つ各 JMS 接続エイリアスに対し、1 つのタスクをスケジュールします。Integration Server はサーバログメッセージを Info レベル、Debug レベル、および Trace レベルとして書き込み、トリガーの再開始タスクの進捗を示します。再開始タスクには、JMS 接続エイリアスを開始するのに使用されたものとは別のスレッドが使用されるため、JMS トリガー再開始の試行は接続エイリアスの起動の完了を阻害しません。

Integration Server は、JMS トリガーを開始するためのトリガーの再開始タスクを試行する間隔、および試行間にタスクが待機する間隔を制御するサーバ設定パラメータを提供します。

- watt.server.jms.trigger.startupFailure.restartTaskRetryCount.JMS 接続エイリアスの起動時に開始に失敗した JMS トリガーを開始するために、トリガー再開タスクが実行する最大再試行回数を指定します。トリガーによって使用される JMS 接続エイリアスの起動時に、Integration Server は JMS トリガーに対して最初の開始を試行します。デフォルトの再試行回数は 6 回です。

- watt.server.jms.trigger.startupFailure.restartTaskRetryInterval.JMS 接続エイリアスの起動時に開始に失敗した JMS トリガーを再開するために、トリガーの再開タスクが試行間に待機する秒数を指定します。正の整数を指定する必要があります。デフォルトは 30 秒です。

JMS トリガーが開始され、実行されている場合、JMS トリガーは JMS プロバイダからのメッセージを受信中に例外をキャッチすることがあります。通常、これは JMS プロバイダが使用できない場合に発生します。JMS プロバイダが使用できない場合、Integration Server は JMS トリガーを無効にします。JMS 接続エイリアスも最終的に失敗します。接続の失敗のため JMS 接続エイリアスが停止すると、Integration Server は 20 秒ごとに JMS 接続エイリアスの再開始を試みます。JMS プロバイダへの接続が復元されると、JMS 接続エイリアスは再開され、Integration Server では JMS 接続エイリアスを使用するすべての JMS トリガーの開始が試行されます。

JMS トリガーが無効になっても、JMS 接続エイリアスがすぐに回復し停止しないことがあります。JMS 接続エイリアスは接続エラーが原因で失敗していないため、Integration Server では JMS 接続エイリアスの再開を試行しません。つまり、Integration Server ではエイリアスの再開後、JMS トリガーの再開は自動的に試行されません。ここで、watt.server.jms.trigger.retryOnConsumerError が true に設定されている場合、Integration Server では JMS トリガーの開始を自動的に試行します。JMS 接続エイリアスが実行されており、JMS トリガーが有効である限り、Integration Server では JMS トリガーの開始を無期限に試行します。

WS エンドポイントトリガーについて

WS (Web サービス) エンドポイントトリガーは、設定オプションが制限されている SOAP-JMS トリガーです。WS エンドポイントトリガーは、プロバイダ Web サービスエンドポイントエイリアスに関連付けられ、このエイリアスが割り当てられているプロバイダ Web サービス記述子のリスナーとして動作します。通常、WS エンドポイントトリガーは、webMethods API Gateway に展開された仮想サービスで使用されます。

WS エンドポイントトリガーを使用する場合は、以下の点に留意してください。

- Integration Server は、WS エンドポイントトリガーを作成するときに、WS Endpoint Trigger: *aliasName* という命名規則を使用します。ここで、*aliasName* はプロバイダ Web サービスエンドポイントエイリアス名です。
- Integration Server は、WS エンドポイントトリガーを設定ファイル *Integration Server_directory* /instances/*instance_name*/config/endpoints/providerJMS.cnf に保存します。
- WS エンドポイントトリガーは、プロバイダ Web サービスエンドポイントエイリアスに関連付けられています。[**WS エンドポイントトリガー**] を JMS トリガーとして選択して、プロバイダ Web サービスエンドポイントエイリアスで使用する場合、Integration Server はエイリアス作成の一部として WS エンドポイントトリガーを作成します。エイリアスを削除すると、WS エンドポイントトリガーも削除されます。
- WS エンドポイントトリガーを作成するときに、逐次処理や JMS 接続エイリアスなどの一部のデフォルトプロパティを設定します。Integration Server Administrator を使用してデフォルト値を変更できます。

  注:

WS エンドポイントトリガーは SOAP-JMS トリガーですが、WS エンドポイントトリガーには SOAP-JMS トリガーのプロパティと機能のサブセットが含まれます。たとえば、重複抑制処理、致命的なエラーの処理、一時的なエラーの処理は SOAP-JMS トリガーでのみ使用できます。

- Integration Server が WS エンドポイントトリガーを作成すると、Integration Server はプレースホルダの宛先を割り当てます。宛先名の形式として wseQueue_*aliasName* を使用します。ここで、*aliasName* はプロバイダ Web サービスエンドポイントエイリアス名です。この名前のキューが存在しない場合は、WS エンドポイントトリガーがサブスクライブする実際の宛先を指定する必要があります。
- Integration Server が WS エンドポイントトリガーを作成すると、Integration Server は WS エンドポイントトリガーを無効な状態のままにします。JMS プロバイダからメッセージを受信するには、このトリガーを有効にする必要があります。JMS トリガーの有効化の詳細については、913 ページの「JMS トリガーのステータスと状態について」を参照してください。
- WS エンドポイントトリガーの編集は、Integration Server Administrator でのみ行うことができます。Designer は WS エンドポイントトリガーの編集には使用できません。
- Integration Server によって、すべての WS エンドポイントトリガーの実行ユーザは、Administrator に設定されます。

WS エンドポイントトリガーの編集

Integration Server がプロバイダ Web サービスエンドポイントエイリアスの作成の一部として WS エンドポイントトリガーを作成すると、Integration Server は他のプロパティのトリガープロパティおよびプレースホルダの一部にデフォルト値を使用します。Integration Server Administrator を使用して、デフォルト値およびプレースホルダの値を変更できます。

WS エンドポイントトリガーを編集するときは、次のガイドラインに従ってください。

- JMS プロバイダとの接続を取得したり、そこからメッセージを受信するときに Integration Server が使用する JMS 接続エイリアスが既に存在している必要があります。JMS 接続エイリアスを WS エンドポイントトリガーに割り当てるときに有効化する必要はありませんが、実行する WS エンドポイントトリガーに対して JMS 接続エイリアスが実行時に有効になっている必要があります。
- JMS 接続エイリアスのトランザクションタイプでは、WS エンドポイントトリガーをトランザクションに使用するかどうか (つまり、トランザクションの一部としてメッセージを受信して処理するかどうか) を決定します。
- WS エンドポイントトリガーがサブスクライブする宛先はキューである必要があります。
- [**実行スレッドの最大数**] に 1 より大きい値を設定できるのは、同時実行トリガーのみです。
- [**接続数**] に 1 より大きい値を指定するには、以下を満たしている必要があります。
  - 指定された JMS 接続エイリアスが、その接続エイリアスを使用する JMS トリガーごとに新しい接続を作成するように設定されていること。詳細については、319 ページの「JMS 接続エイリアスの複数接続の許可」を参照してください。
  - WS エンドポイントトリガーが、並行処理用に設定されていること。

WS エンドポイントトリガーを編集するには

1. Integration Server Administrator を開きます。
2. [**メッセージング**] > [JMS トリガー] に移動します。
3. [**個別 SOAP JMS トリガー制御**] で、編集する WS エンドポイントトリガーを選択します。
4. [**メッセージング**] > [JMS トリガー] > [詳細] > [*triggerName*] ページで、[**トリガーの編集**] をクリックします。
5. [**JMS 接続エイリアス**] リストで、JMS プロバイダからメッセージを受信するときに WS エンドポイントトリガーが使用する JMS 接続エイリアスを選択します。
6. [**宛先名**] フィールドに、WS エンドポイントトリガーがメッセージを受信するキューの名前を入力します。
7. [**処理モード**] の横で、以下のいずれかを選択します。

選択項目	目的
[逐次]	Integration Server がトリガーで受信したメッセージを順番に処理するように指定します。
[並行]	Integration Server がこのトリガーに対する複数のメッセージを一度に処理するように指定します。

8. 並行処理を選択した場合は、[**実行スレッドの最大数**] フィールドで、Integration Server が並行して処理できるメッセージの最大数を指定します。
9. 並行処理を選択した場合は、[**接続数**] フィールドで、webMethods Broker に対して WS エンドポイントトリガーが接続する回数を指定します。

> 注:
> [**接続数**] の値は、[**実行スレッドの最大数**] の値以下である必要があります。

10. [**変更内容の保存**] をクリックします。

48 MQTT トリガー

- MQTT トリガーの管理の概要 932
- MQTT トリガーの状態とステータス 932
- MQTT トリガーに対する Integration Server セッションの再利用の設定 934
- MQTT トリガーの開始の自動再試行 934

MQTT トリガーの管理の概要

Integration Server には、MQTT トリガーおよびトリガーが使用するリソースを管理するための機能が用意されています。

- トリガー状態を変更する。MQTT トリガーがメッセージを受信するのか、処理するのか、またはその両方なのかを決定する。
- Integration Server が MQTT トリガーのセッションを再使用するよう設定する。
- 最初の起動に失敗した MQTT トリガーに対して Integration Server が自動的に再開始を試行するよう設定する。

MQTT トリガーの状態とステータス

各 MQTT トリガーには状態とステータスがあります。状態は、トリガーの設定が有効、無効、または一時停止中のいずれであるかを示します。ステータスにはトリガーの現在の状態が反映されます。MQTT トリガーのステータスは [Running] または [Not Running (*reason*)] を表示することができます。*reason* は、[Not Running (Trigger Disabled)]、[Not running (Connection Alias Failure)] など、トリガーが実行されていない理由を識別します。たとえば、MQTT トリガーは、状態が [有効]、ステータスが [Not running (Trigger Failure)] になることがあります。

次の表は、MQTT トリガーに対して可能な状態を示しています。

トリガーの状態	説明
有効	MQTT トリガーは実行中で、MQTT サーバに接続されています。Integration Server は、MQTT トリガーに対してメッセージの抽出と処理を行います。
無効	MQTT トリガーは停止されています。MQTT トリガーは MQTT サーバに接続されていません。Integration Server は、MQTT トリガーに対してメッセージの抽出も処理も行いません。MQTT トリガーは、有効化されるまでこの状態のままです。 MQTT トリガーは一時的に無効な状態になることがあります。これは「無効」の後のアスタリスク (*) で示されます。再試行失敗の発生後に、再試行失敗処理動作である [**トリガーの停止**] が有効になると、MQTT トリガーは一時的に無効になることがあります。再試行失敗処理動作および Integration Server が一時的に無効になった MQTT トリガーを自動的に有効にする際の詳細については、*webMethods Service Development Help*を参照してください。
一時停止	MQTT トリガーは実行中で、MQTT サーバに接続されていますが、新しいメッセージの抽出または処理を行っていません。Integration Server は MQTT サーバからのメッセージ抽出を停止し、MQTT トリガー用のローカルトリガーキューからのメッセージプルを停止しますが、ローカルトリガーキューから既に抽出したメッセージの処理は続行します。 Integration Server は、サーバの再起動時または MQTT トリガーを含むパッケージの再ロード時に MQTT トリガーを自動的に有効化します。

MQTT トリガーの状態の変更

Integration Server で 1 つの MQTT トリガーまたはすべての MQTT トリガーの状態を変更できます。

MQTT トリガーの状態を変更するときは、以下の情報に留意してください。

- MQTT トリガーを無効にする場合は、最初に MQTT トリガーを一時停止して、処理中のすべてのトリガースレッドが完了するまで待ちます。その後で、トリガーを無効にします。現在 MQTT トリガーで使用されているスレッドの数は、[**メッセージング**] > [MQTT トリガー] ページの [**現在のスレッド**] 列で確認できます。
- MQTT トリガーを最初に一時停止せずに無効にした場合、トリガーが無効になる前にメッセージ処理が完了しないことがあります。
- 有効な MQTT トリガーを一時停止した場合、Integration Server は次の処理を行います。
  - MQTT サーバからのメッセージの抽出を停止します。
  - ローカルトリガーキューから処理用のメッセージのプルを停止します。
  - トリガーがローカルトリガーキューから既に抽出したメッセージの処理を続行します。
- すべてのメッセージ処理が完了した一時停止中の MQTT トリガーを無効にした場合、Integration Server は次の処理を行います。
  - ローカルトリガーキューを消去します。
  - トリガーがメッセージの抽出に使用するリスナーを停止します。これによって、MQTT トリガーは MQTT サーバから切断されます。
- 有効な MQTT トリガーを無効にした場合 (つまり、最初にトリガーを一時停止していない場合)、Integration Server は次の処理を行います。
  - MQTT サーバからのメッセージの抽出を停止します。
  - ローカルトリガーキューから処理用のメッセージのプルを停止します。
  - メッセージ処理の完了のため、短時間待機します。タイムアウトが経過するまでにメッセージ処理が完了しなかった場合、Integration Server は MQTT トリガー用のリスナーを停止します。Integration Server がリスナーを停止した後、Integration Server は未解決のメッセージの受信を確認できません。MQTT トリガーのサーバスレッド処理メッセージは完了するまで続行しますが、処理完了時の MQTT メッセージの受信確認は失敗します。

    サーバ設定パラメータ watt.server.commonmessaging.trigger.stopRequestTimeOut は、Integration Server が処理の完了まで待機する最大時間を決定します。デフォルトは 3 秒です。
  - MQTT トリガーが一時的エラーの処理のために再試行を待機している場合は、Integration Server はその MQTT トリガーの処理を中断します。その後の再試行は行いません。
  - ローカルトリガーキューを消去します。

- トリガーがメッセージの抽出に使用するリスナーを停止します。これによって、MQTT トリガーは MQTT サーバから切断されます。

MQTT トリガーの状態を変更するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**メッセージング**] > [MQTT トリガー] に移動します。
3. すべての MQTT トリガーの状態を変更する場合は、以下の手順に従います。
   a. [**有効**] 列で、[**すべてを編集**] をクリックします。
   b. [**メッセージング**] > [MQTT トリガー] > [トリガーの状態の編集] ページの [**新規の状態**] リストで、すべての MQTT トリガーに適用させる状態を選択します。
4. 特定の MQTT トリガーの状態を変更する場合は、以下の手順に従います。
   a. 有効化、無効化または一時停止する MQTT トリガーの行にある [**状態**] 列で現在の状態を選択します。
   b. [**メッセージング**] > [MQTT トリガー] > [トリガーの状態の編集] ページの [**新規の状態**] リストで、この MQTT トリガーに適用させる状態を選択します。
5. [**変更内容の保存**] をクリックします。

MQTT トリガーに対する Integration Server セッションの再利用の設定

MQTT トリガーがサービスを呼び出すたびに同じ Integration Server セッションを再利用すると、サーバのパフォーマンスを向上させることができます。セッションを再利用すると、サービスの呼び出しに Integration Server で必要となるオーバーヘッドの量が減少します。

再利用されたセッションでサービスを呼び出すと、それぞれが同じセッション ID を使用します。一意のセッション ID を必要とするサービスではセッションを再利用できません。

すべての MQTT トリガーが Integration Server セッションを再利用するよう設定するには、watt.server.commonmessaging.trigger.reuseSession サーバ設定パラメータを「true」に設定します。デフォルトでは、このパラメータは「false」に設定されます。

MQTT トリガーの開始の自動再試行

MQTT 接続エイリアスの起動時に、Integration Server は、メッセージの抽出にエイリアスを使用する有効な MQTT トリガーも開始します。正常に起動された後 MQTT 接続エイリアスに障害が発生するか、利用できなくなった場合、接続エイリアスが正常に再起動した後、Integration Server は有効な MQTT トリガーを再開始しようとします。エイリアスの起動または再起動プロセスの間、MQTT トリガーの開始 (または再開始) の試行は失敗することがあります。たとえば、

MQTT トリガーと MQTT サーバのどちらか一方または両方が適切に設定されていない場合、トリガーは開始されません。

MQTT 接続エイリアスの起動 (または再起動) 時に MQTT トリガーが開始しない場合、Integration Server は MQTT トリガーの開始を再試行するためのタスクをスケジュールし、その後 MQTT 接続エイリアスの起動に関連するアクティビティを続行します。独自のスレッドで実行するトリガーの再開始タスクは、設定された間隔で、失敗した MQTT トリガーの再開始を試みます。デフォルトでは、タスクは 30 秒ごとに実行されます。MQTT 接続エイリアスが無効になったときに失敗したすべてのトリガーが正常に開始した後、または最大再試行回数に達した後、タスクは完了します。最大再試行回数に達しても MQTT トリガーが開始しない場合、残りの MQTT トリガーの設定の問題を調査する必要があります。設定の問題が解決されると、Integration Server を使用して MQTT トリガーを手動で再開始することができます。

Integration Server は、エイリアスが正常に起動した際に開始されないトリガーを持つ各 MQTT 接続エイリアスに対し、1 つのタスクをスケジュールします。Integration Server はサーバログメッセージを Info レベル、Debug レベル、および Trace レベルとして書き込み、トリガーの再開始タスクの進捗を示します。再開始タスクには、MQTT 接続エイリアスを起動するのに使用されたものとは別のスレッドが使用されるため、MQTT トリガー再開始の試行は接続エイリアスの起動の完了を阻害しません。

Integration Server は、MQTT トリガーを開始するためのトリガーの再開始タスクを試行する間隔、および試行間にタスクが待機する間隔を制御するサーバ設定パラメータを提供します。

- watt.server.commonmessaging.trigger.restartTaskRetryCount。MQTT 接続エイリアスの起動時に開始に失敗した MQTT トリガーを開始するために、トリガー再開始タスクが実行する最大再試行回数を指定します。トリガーによって使用される MQTT 接続エイリアスの起動時に、Integration Server は MQTT トリガーの最初の開始を試行します。デフォルトの再試行回数は 6 回です。
- watt.server.commonmessaging.trigger.restartTaskRetryInterval。MQTT 接続エイリアスの起動時に開始できなかった MQTT トリガーを再開始するためのトリガーの再開始タスクが試行間に待機する秒数を指定します。正の整数を指定する必要があります。デフォルトは 30 秒です。

49 重複抑制処理を設定したドキュメント履歴データベースの使用

- 概要 938
- ドキュメント履歴データベースからの期限切れエントリの削除 938
- 重複抑制処理の統計情報の表示 939
- 重複抑制処理の統計情報の消去 939

概要

ドキュメント履歴データベースには、webMethods messaging triggerによって処理された保証付きドキュメントおよび JMS トリガーによって処理された永続メッセージが保持されます。この機能は、ドキュメント履歴が重複検出の一環として使用されるように設定しているトリガーに対してのみ有効です。トリガーサービスが実行を開始したとき、および (成功、失敗にかかわらず) サービスの実行が完了したときに、Integration Server によってドキュメント履歴データベースにエントリが追加されます。

webMethods messaging trigger または JMS トリガーの重複抑制処理の詳細については、『*webMethods Service Development Help*』を参照してください。

ドキュメント履歴データベースからの期限切れエントリの削除

ドキュメント履歴データベースを管理しやすいサイズに保つために、Integration Server は期限切れの行をデータベースから定期的に削除します。トリガーの [**履歴を廃棄するまでの時間**] プロパティの値によって、ドキュメント履歴データベースが処理済みドキュメントまたはメッセージのエントリを保持する期間が決まります。

Integration Server には、期限切れのエントリをデータベースから削除するスケジュール済みサービスが用意されています。デフォルトでは、メッセージ履歴スイーパタスクは 10 分ごとに実行されます。サービスの実行頻度は変更できます。スケジュール済みサービスを編集する方法については、763 ページの「サービスのスケジュール」を参照してください。

注:
メッセージ履歴スイーパタスクの初期実行頻度は、watt.server.idr.reaperInterval プロパティによって決まります。Integration Server とドキュメント履歴データベースとの通信に使用する JDBC 接続プールを定義してから、スケジュール済みサービスを編集して実行間隔を変更します。

また、データベースからいつでもすべての期限切れエントリを消去できます。詳細については、938 ページの「ドキュメント履歴データベースからの期限切れエントリの消去」を参照してください。

ドキュメント履歴データベースからの期限切れエントリの消去

期限切れエントリを削除するために、スケジュールされている次のメッセージ履歴スイーパタスクの実行まで待つのが好ましくない場合は、Integration Server Administrator を使用してすべての期限切れエントリを削除することができます。

期限切れのエントリをドキュメント履歴データベースから消去するには

1. Integration Server Administrator を開きます。
2. [**設定**] > [リソース] に移動します。
3. [**重複抑制処理の統計情報**] をクリックします。

4. [**期限切れドキュメント履歴の削除**] をクリックします。

重複抑制処理の統計情報の表示

Integration Server Administrator を使用すると、JMS トリガーで受信したインダウトメッセージまたは重複メッセージの履歴を表示できます。Integration Server Administrator には、重複抑制処理を設定したトリガーで受信した重複メッセージまたはインダウトメッセージの名前、UUID (Universally Unique Identifier : 汎用一意識別子) および状態が表示されます。

Integration Server は、重複抑制処理の統計情報をメモリに保存します。Integration Server が再起動すると、統計情報はメモリから削除されます。

重複抑制処理の統計情報を表示するには

1. webMethods Integration Server を起動して、Integration Server Administrator を開きます。
2. [**設定**] > [リソース] に移動します。
3. [**重複抑制処理の統計情報**] をクリックします。

重複抑制処理の統計情報の消去

Integration Server Administrator を使用して、インダウトドキュメントおよび重複ドキュメントの統計情報を削除することができます。

重複抑制処理の統計情報を消去するには

1. webMethods Integration Server を起動して、Integration Server Administrator を開きます。
2. [**設定**] > [リソース] に移動します。
3. [**重複抑制処理の統計情報**] をクリックします。
4. [**すべての重複またはインダウトドキュメントの統計情報を消去**] をクリックします。

50 Integration Server での XA トランザクション管理

- XA トランザクション管理の概要 942
- Integration Server での XA オプションの設定 946
- 手動でのトランザクションの解決 949

XA トランザクション管理の概要

トランザクションとは論理的な作業単位であり、全体的に成功するか、あるいは、まったく影響を及ぼさない、一連の異なるプロセスで構成されています。

トランザクションで実行されている作業は、多様なプラットフォームで発生したり、さまざまなベンダーからのさまざまなリソースに影響したりすることがあるため、X/Open という団体が DTP (Distributed Transaction Process : 分散トランザクション処理) モデルと XA インタフェースを開発しました。

DTP モデルでは、以下の間の通信が定義されています。

- データベースなどのリソース。
- データベースサーバなどのリソースマネージャ。リソースマネージャは、共有リソースへのアクセスを提供します。
- トランザクションマネージャ。トランザクションマネージャは、リソースマネージャを通じて、リソースに関するすべてのトランザクションを調整および管理します。Integration Server には、webMethods Adapter for JDBC や webMethods Adapter for JMS が開始したデータベーストランザクションを調整および管理するトランザクションマネージャサブシステムが含まれています。

XA インタフェースには、リソースとトランザクションマネージャとの間で行われるトランザクションの調整、コミットメント、回復に関するプロトコルが記述されています。Integration Server は XA プロトコルをサポートしており、2 フェーズコミット (2PC) と呼ばれるトランザクションプロトコルを使用してトランザクションを管理します。

2PC の最初のフェーズでは、Integration Server (特に Transaction Manager) は、トランザクションをコミットできる状態であるかどうかを、トランザクションに参加しているリソースに問い合わせます。2 番目のフェーズでは、以下のいずれかが行われます。

- すべてのリソースが、コミットできる状態であると応答する。Integration Server は、トランザクションをコミットするようにリソースに指示します。
- 1 つまたは複数のリソースが、コミットできない状態であると応答する。Integration Server は、トランザクションをコミットするための準備をロールバックするようにすべてのリソースに指示します。

Integration Server ではトランザクションの状態がどのようにパーシストされるのか

トランザクションの開始時に、Integration Server は XID という一意のトランザクション ID を作成します。Integration Server は、この XID と、トランザクションのグローバル状態を「XA 回復ストア」というパーシスタントストアに格納します。トランザクションに対するその後の各アクションの開始時に、Integration Server はトランザクションのグローバル状態と、トランザクションに参加している各リソースの状態を XA 回復ストアに格納します。Integration Server が異常終了すると、Integration Server は状態情報を XA 回復ストアから取得し、未完了トランザクションがある場合はそれを解決しようとします。

Integration Server が状態情報を格納するには、以下の条件が満たされている必要があります。

- watt.server.transaction.xastore.performXALogging サーバ設定パラメータは true に設定されている必要がある。Integration Server が XA 回復ストアにトランザクション情報を書き込むのは、このプロパティが true (デフォルト値) に設定されているときだけです。
- トランザクションに複数のリソースが関与し、すべてのリソースが XA に対応している (すなわちリソースが JTA および XA の規格をサポートしており、準備が完了したか、またはヒューリスティックにコミットされたトランザクションの永続的な (パーシスタント) レコードを保持している)。
- トランザクションが XA トランザクションとして定義されている。たとえば、トランザクションに webMethods Adapter for JDBC が関与している場合、そのトランザクションは、このアダプタがリソースに接続するときに XA トランザクションとして定義されます。

注:
信頼性や回復性を向上するほとんどの機能と同じように、この機能によって XA トランザクションの処理に関するオーバーヘッドが増加することがあります。

Integration Server では未完了トランザクションがどのように解決されるのか

トランザクションの処理中に Integration Server が異常終了した場合、これらのトランザクションは完了されません。Integration Server は、再起動時に未完了トランザクションのリストを XA 回復ストアから抽出し、Integration Server がそのリストに記録した各トランザクションの最新の状態に基づいて、Integration Server は以下のようにして各トランザクションを解決しようとします。

状態	Integration Server の対応
リソースがコミットプロセスを開始し、少なくとも 1 つのリソースがトランザクションをコミットした	その他のリソースがコミットするように試みる
リソースがトランザクションのコミット準備を完了したが、コミットプロセスを開始していなかった	すべてのコミット準備をロールバックするようにすべてのリソースに指示する
リソースがコミットプロセスを開始したが、どのリソースもトランザクションをコミットしていなかった	すべてのコミット準備をロールバックするようにすべてのリソースに指示する
リソースがトランザクションのロールバックを開始したが、完了していなかった	すべてのコミット準備をロールバックするようにすべてのリソースに指示する
Integration Server が、トランザクションをコミットできる状態であるかどうかをリソースに問い合わせていなかった	トランザクションを無視し、その XID を XA 回復ストアから削除する
リソースがトランザクションのコミットまたはロールバックを完了していた	トランザクションを無視し、その XID を XA 回復ストアから削除する

未完了トランザクションを Integration Server が解決しようとしているときにエラーが発生した場合、Integration Server は指定されている時間だけ待機してから、もう一度解決しようとします。Integration Server は、指定の最大回復時間が経過するまでこの回復動作を継続します。これらの値を設定する方法の詳細については、949 ページの「XA サーバパラメータの設定」を参照してください。

Integration Server が解決を試行している間は、新しい XA トランザクションは妨害されない状態になったままになります。

Integration Server は、未完了トランザクションをすべて解決できるわけではありません。たとえば Integration Server は、以下の場合にはトランザクションを解決することができません。

- Integration Server が異常終了した後で、リソースでトランザクションのコミットまたはロールバックをリソース管理者が強制的に実行した。
- トランザクションに 1PC (1 フェーズコミット) リソースが含まれており、参加リソースが XA に対応しているトランザクションのみのステータスを Integration Server が格納している。
- 指定された最大回復時間内に回復処理の試行を繰り返した後、Integration Server がリソースに接続できない (たとえば、トランザクションが webMethods Adapter for JDBC に影響するが、リソースへのアダプタの接続が存在しないか、変更されているため)。

このような場合は、未完了トランザクションを手動で解決する必要があります。

未解決の XA トランザクションについて

手動で解決する必要がある XA トランザクションは、Integration Server の [**設定**] > [リソース] > [XA 手動回復] ページに表示されます。Integration Server では、未解決のトランザクションはテーブルに一覧表示され、それぞれについて以下の情報が表示されます。

列	説明
[XID]	トランザクションの一意の XID。Integration Server はトランザクションの開始時に XID を作成し、XA 回復ストアに書き込みます。また、Integration Server は参加リソースにも XID を渡すため、これらのリソースにもこの情報が格納されます。
[グローバル 2PC 状態]	Integration Server が終了する前のトランザクションのグローバル状態。JTA 規格に記述されている javax.transaction.Status インタフェースで、状態がグローバル状態にマッピングされる場合、そのマッピングは以下のように表示されます。

[状態]	説明
TR_PREPARE_BEGIN	STATUS_PREPARING
TR_PREPARE_RESOURCE	
TR_PREPARE_RESOURCE_END	
TR_PREPARE_END	STATUS_PREPARED

列	説明	
	TR_COMMIT_BEGIN	STATUS_COMMITTING
	TR_COMMIT_RESOURCE	
	TR_COMMIT_RESOURCE_END	
	TR_ROLLBACK_BEGIN	STATUS_ROLLING_BACK
	TR_ROLLBACK_RESOURCE	
	TR_ROLLBACK_RESOURCE_END	
	TR_ROLLBACK_END	STATUS_ROLLED_BACK
	TR_ROLLBACK_ONLY	MARKED_ROLLBACK
	TR_FORGET_RESOURCE	
	TR_FORGET_RESOURCE_END	
	TR_COMPLETED	
	TR_RECOVERY	Integration Server はトランザクションを解決しようとしています。
	TR_UNDEFINED	STATUS_UNKNOWN
[エラーメッセージ]	トランザクションのグローバル状態を XA 回復ストアに格納する前に Integration Server がサーバログに書き込むエラーメッセージ。	
[試行済みの回復アクション]	トランザクションを解決するために、Integration Server が行ったアクション。[**グローバル2PC状態**] が「TR_COMMIT_BEGIN」である場合、Integration Server はトランザクションをコミットしようとします。グローバル状態がその他の値である場合は、Integration Server はトランザクションをロールバックしようとします。	

注:
間隔をおいてページをリフレッシュして、すべての未完了トランザクションが一覧表示されるようにしてください。Integration Server が未完了トランザクションを解決しようとしたが、解決できなかったとします。この場合、Integration Server が未完了トランザクションを解決しようとしている間、このトランザクションは一覧表示されませんが、後でページをリフレッシュすると、一覧に表示されます。

未解決の XA トランザクションの詳細

未解決の各 XA トランザクションについて、参加リソース、各リソースでのトランザクションの状態、リソースでのトランザクションの推定状態などの、詳細な情報を表示することができます。[**設定**] > [リソース] > [XA 手動回復] ページで、未解決のトランザクションの XID をクリッ

クすると、トランザクションに関連している各リソースの以下の情報が Integration Server Administrator に表示されます。

列	説明
[XA リソース]	リソースの完全修飾名。
[XID の有無]	トランザクションの XID がリソースに存在するかどうかを示します。

値	説明
はい	XID がリソースに存在します。
いいえ	XID は存在しません。
[不明]	Integration Server は XID がリソースに存在するかどうかを特定できませんでした。

[状態]	リソースでのトランザクションの現在の状態。これは、[**グローバル 2PC 状態**] リストの値と同じです。グローバル 2PC 状態の一覧については、944 ページの「未解決の XA トランザクションについて」の表を参照してください。
[推定状態]	[**XID の有無**] および [**状態**] の値に基づいて推定される、リソースでのトランザクションの状態。有効な組み合わせに基づいて、状態は以下のようになります。

[XID の有無]	[状態]	[推定状態]
はい	すべて	準備完了、またはヒューリスティックアクションが行われた
いいえ	TR_ROLLBACK_RESOURCE_END	ロールバックされている
いいえ	TR_FORGET_RESOURCE_END	無視されている
いいえ	TR_ROLLBACK_RESOURCE_END および TR_FORGET_RESOURCE_END を除く、すべての状態	コミットされている

トランザクションを手動で解決する方法については、949 ページの「手動でのトランザクションの解決」を参照してください。

Integration Server での XA オプションの設定

Integration Server を使用すると、以下の XA のオプションを設定することができます。

- XA トランザクション回復の有効化または無効化

- XA 回復ストアの場所および初期サイズ
- 以下のものを決定するサーバパラメータ
  - トランザクションの解決を試行してから、次に試行するまで Integration Server が待機する時間の長さ
  - トランザクションを解決するのに許容されている最長時間
  - Integration Server がトランザクションの状態とその参加リソースを (ストアが損傷しているなどの理由で) XA 回復ストアに格納できない場合に、トランザクションを続行するかどうか
  - XA 回復ログファイル内の一意の XA トランザクションの最大数
  - Integration Server が XA 回復ストアにトランザクション情報を書き込むかどうか

次に、これらのオプションの設定の詳細について説明します。

XA トランザクション回復の有効化または無効化

XA トランザクション回復を有効化または無効化できます。XA トランザクション回復を無効化すると、Integration Server は未完了トランザクションを自動的に回復しません。また、Integration Server Administrator を使用して未完了トランザクションを手動で回復することはできません。

XA トランザクションの信頼性および回復性と引き換えに、処理パフォーマンスを向上させたい場合は、XA トランザクション回復を無効化することができます。デフォルトでは、Integration Server でXA トランザクション回復は有効化されています。

XA トランザクション回復を有効化または無効化するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**設定**] > [リソース] > [XA 手動回復] をクリックします。
3. 次のいずれかの手順に従います。
   - 現在有効になっている XA トランザクション回復を無効にする場合は、[**XA トランザクション回復を無効にする**] をクリックします。Integration Server Administrator で、[**未解決の XA トランザクション**] テーブルが非表示になります。
   - 現在無効になっている XA トランザクション回復を有効にする場合は、[**XA トランザクション回復を有効にする**] をクリックします。Integration Server Administrator に、[**未解決の XA トランザクション**] テーブルが表示されます。

     ヒント:
     サーバパラメータ watt.server.jca.transaction.writeRecoveryRecord を使用して、XA トランザクション回復を有効化または無効化することも可能です。サーバパラメータの設定の詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

XA 回復ストアの設定

XA 回復ストアは、トランザクションの XID とグローバル状態が格納されているパーシスタントストアです。XA 回復ストアの場所と、起動時のファイルの初期サイズを指定することができます。

XA 回復ストアを設定するには

1. [**設定**] > [リソース] > [ストアの設定] をクリックします。Integration Server Administrator の [**XA 回復ストア**] セクションに、XA 回復ストアの場所と初期サイズの現在の設定が表示されています。

2. [**XA 回復ストアの設定の編集**] をクリックします。

3. [**ストアの場所**] フィールドで、XA 回復ストアのファイルを格納する、ファイルシステム内のディレクトリへの相対パスまたは絶対パスを入力します。デフォルトの場所を以下に示します。

   Integration Server_directory \instances*instance_name*\XAStore

   重要:
   指定したディレクトリに対する書き込みアクセス権を持っていることを確認してください。また、ディレクトリ名にはオペレーティングシステムによって不正と見なされる文字を使用しないでください。

4. [**ストアの初期サイズ (MB)**] フィールドで、XA 回復ストアのファイルの初期サイズをメガバイト単位で入力します。デフォルトは 10 MB に設定されています。

   注:
   Integration Server がインストールされているコンピュータにデフォルトのドキュメントストア、トリガードキュメントストア、および XA 回復ストアの初期サイズを保存するだけの十分な空きディスク容量があることを確認してください。

5. [**変更内容の保存**] をクリックします。

   次の再起動時に、Integration Server は新しい XA 回復ストアのファイルを新しい場所に作成し、そのファイルへの書き込みを開始します。Integration Serverまた、そのファイルには、新しい初期サイズが使用されます。

   注:
   XA 回復ストアの初期サイズと場所を指定するのに加え、watt.server.transaction.xastore.maxTxnPerFile を使用して、XA 回復ログファイル内の一意の XA トランザクションの最大数を指定できます。このサーバパラメータの詳細については、以下を参照してください。 1095 ページの「サーバ設定パラメータ」

XA サーバパラメータの設定

Integration Server には、XA トランザクションおよび XA トランザクション回復の以下のサーバパラメータが用意されています。

watt.server.transaction.recovery.sleepInterval
Integration Server が未完了トランザクションを解決しようとしているときにエラーが発生した場合に、次に解決を試行するまで Integration Server が待機する時間を指定します。

watt.server.transaction.recovery.abandonTimeout
トランザクションを解決するための最大回復時間を指定します。Integration Server は、最大回復時間になるまでトランザクションを解決しようとします。

watt.server.jca.transaction.rollbackOnWriteFailure
Integration Server がトランザクションの状態とその参加リソースを (ストアが損傷しているなどの理由で) XA 回復ストアに格納できない場合に、Integration Server がトランザクションを続行するか、ロールバックするかを指定します。このパラメータを「false」に設定するには、多少のリスクが伴います。この場合、Integration Server が異常終了しても XA 回復ストアに状態が保存されないため、Integration Server は未完了トランザクションを解決できないほか、未完了トランザクションを手動で解決することもできません。

watt.server.transaction.xastore.maxTxnPerFile
XA 回復ログファイル内の一意の XA トランザクションの最大数を指定します。XA 回復ログファイルがトランザクションの最大数に達すると、Integration Server は新しいファイルを作成します。デフォルトは 2000 トランザクションです。

watt.server.transaction.xastore.performXALogging
Integration Server が XA 回復ストアにトランザクション情報を書き込むかどうかを指定します。true に設定すると、Integration Server は各 XA トランザクションの状態と進捗に関する情報をログに記録します。false に設定すると、Integration Server は XA トランザクション情報のロギングをスキップします。デフォルトは「true」です。

> **重要:**
> このプロパティを false に設定した場合、Integration Server はトランザクションの処理中に XA 回復ストアに情報を記録しないため、トランザクションの回復は不可能になります。Integration Server が未完了トランザクションを自動的に解決するようにするか、未完了トランザクションを手動で解決する場合、Integration Server は XA ロギングを実行する必要があります。

上記およびその他のサーバパラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

手動でのトランザクションの解決

Integration Server がトランザクションを解決できない場合は、手動での解決を試みます。トランザクションを手動で解決するには、参加リソースを把握し、トランザクションが一貫した状態を保つような方法で作業する必要があります。たとえば、参加リソースの 1 つだけがトランザクションをコミットした場合、その他の参加リソースもコミットするようにすることができます。

トランザクションを手動で解決するときは、解決法そのものはトランザクションではありません。つまり、各参加リソースと、それに対して実行するアクションは、新しい 2PC トランザクションに参加しません。このため、アクションを行った結果、参加リソースが一貫した状態になるようにする必要があります。

XA トランザクションを手動で解決するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**設定**] > [リソース] > [XA 手動回復] をクリックします。Integration Server Administrator に、未解決の XA トランザクションがすべて表示されます。未解決の各トランザクションについて表示される情報の詳細については、944 ページの「未解決の XA トランザクションについて」を参照してください。

   注:
   間隔をおいてページをリフレッシュして、すべての未完了トランザクションが一覧表示されるようにしてください。Integration Server が未完了トランザクションを解決しようとしたが、解決できなかったとします。この場合、Integration Server が未完了トランザクションを解決しようとしている間、このトランザクションは一覧表示されませんが、後でページをリフレッシュすると、一覧に表示されます。

3. [**XID**] 列で、解決するトランザクションの XID をクリックします。そのトランザクションに関連しているリソースの詳細情報が Integration Server Administrator に表示されます。参加している各リソースについて表示される情報の詳細については、945 ページの「未解決の XA トランザクションの詳細」を参照してください。

4. トランザクションを削除する場合は、[**トランザクションの削除**] リンクをクリックします。トランザクションを削除すると、そのトランザクションは XA 回復ストアから削除されます。

   リソースを直接操作してトランザクションを解決するなどの理由で、トランザクションを削除するのに Integration Server Administrator を使用しない場合は、単にトランザクションを削除してもかまいません。

5. Integration Server Administrator を使用してトランザクションを解決する場合は、[**実行するアクション**] 列で以下のいずれかを選択します。

目的	選択項目
リソースでトランザクションをコミットする	**[コミット]**
リソースでトランザクションをロールバックする	**[ロールバック]**
リソース管理者がトランザクションをヒューリスティックにコミットまたはロールバックしたため、XID をリソースから削除する	**[無視]**

目的	選択項目
リソース管理者が、既にリソースで修正アクションを行ったため、アクションを行う必要がないか、長期間にわたってリソースが停止している	**[何もしない]**

注:
[**実行するアクション**] 列に、リソースの [**状態**] および [**XID**] の値の組み合わせに基づいて、各リソースに対して実行可能なアクションが一覧表示され、最も論理的なアクションがデフォルトで選択されます。

6. [**アクションの実行**] をクリックします。Integration Server Administrator の [**XA 手動回復**] ページに戻り、トランザクションが未解決トランザクションの一覧から削除されます。

   Integration Server はリソースからエラーを受信して表示することがあります。エラーは以下の理由で発生することがあります。

   - リソースが停止していたなどの理由により、リソースが Integration Server に接続されていなかった。
   - 2PC トランザクションの情報が失われていたなどの理由により、リソースがトランザクションを認識していなかった。
   - リソースが例外をスローした。
   - トランザクションが webMethods アダプタに影響するが、Integration Server から、リソースを指していたアダプタの接続ノードが削除または変更されたため、Software AG Designer がそのリソースを特定できない。

   リソースに用意されているツールを使用して、トランザクションを一貫した状態にする必要があることがあります。

51 Integration Server のコンテンツハンドラ

- Integration Server がコンテンツハンドラを使用する方法 954
- Integration Server が HTTP 要求のためにコンテンツハンドラを選択する方法 954
- Integration Server が FTP 要求のためにコンテンツハンドラを選択する方法 954
- HTTP および FTP 要求のためのコンテンツハンドラ 954
- HTTP 応答のコンテンツハンドラについて 960
- FTP 応答のコンテンツハンドラについて 962

Integration Server がコンテンツハンドラを使用する方法

クライアントで HTTP または FTP 要求がサブミットされると、Integration Server はコンテンツハンドラを使用して要求の内容を解析し、その結果をターゲットサービスに渡します。Integration Server は HTTP または FTP 応答の構築時にもコンテンツハンドラを使用します。

Integration Server が HTTP 要求のためにコンテンツハンドラを選択する方法

受信 HTTP 要求に使用するコンテンツハンドラを決定するために、Integration Server は要求ヘッダーのコンテンツタイプヘッダーフィールドを確認して、そのコンテンツタイプ用に登録されているコンテンツハンドラを使用します。たとえば、要求ヘッダーで Content-Type=text/html と指定されている場合、Integration Server は text/html コンテンツハンドラとして登録されているコンテンツハンドラを探します。

Integration Server が FTP 要求のためにコンテンツハンドラを選択する方法

受信 FTP 要求のために、どのコンテンツハンドラを使用するかを決定するために、Integration Server 上の FTP 受信待機ポートは、要求のファイル拡張子に基づいてコンテンツハンドラを選択します。 *Integration Server_directory* /instances/*instance_name*/lib/mime.types ファイルには、ファイル拡張子とコンテンツタイプのマッピングが保存されています。

[**設定**] > [**リソース**] > [**MIME タイプの設定**] を選択することで、Integration Server Administrator から lib/mime.types ファイルのマッピングを編集できます。lib/mime.types ファイルのマッピングを編集する方法の詳細については、『*webMethods Service Development Help*』を参照してください。

HTTP および FTP 要求のためのコンテンツハンドラ

Integration Server は要求のコンテンツタイプに基づいてコンテンツハンドラを選択し、コンテンツハンドラを使用して要求の内容を解析します。

次の表に、登録されたコンテンツタイプに基づいて、Integration Server が受信 HTTP および FTP 要求のためにどのコンテンツハンドラを使用するかを示します。

Integration Server が使用するコンテンツハンドラ	コンテンツタイプ	解析する対象
ContentHandler_CGI	text/html、および watt.server.default ContentHandler が `false` に設定されている	2CGI コンテンツと書式設定されていないテキスト。 このコンテンツハンドラは、以下の処理を実行します。 ■ 受信入力ストリームをデコードして、コンテンツをキー/値のペア

Integration Server が使用するコンテンツハンドラ	コンテンツタイプ	解析する対象
		に変換します。キーが既に存在している場合、値がキー key+list の形式でパイプライン内のリストに追加されます。 ■ キー/値のペアを HTML の表に書き込みます。
	application /x-www-form-urlencoded	Web フォームから取得された名前/値ペア。
ContentHandler_Default	text/html、および watt.server.default ContentHandler が `true` に設定されている	HTTP 要求からの入力ストリームまたは FTP 要求からのドキュメント。 このコンテンツハンドラは、以下の処理を実行します。 ■ HTTP 要求の場合、ハンドラは、入力ストリームのコンテンツを HTTP 要求から、contentStream という入力ストリーム内のサブミットメソッドに指定されたサービスに渡します。 ■ FTP 要求の場合、ハンドラは、ドキュメントのコンテンツを FTP 要求から、contentStream という入力ストリーム内のサブミットメソッドに指定されたサービスに渡します。 ■ HTTP および FTP の場合、ハンドラはキー/値のペアを HTML の表に書き込みます。
ContentHandler_FlatFile	application/x-wmflatfile	フラットファイルのコンテンツ。 このコンテンツハンドラは、以下の処理を実行します。 ■ ffdata という名前の入力ストリームのサブミットメソッドで指定されたサービスにフラットファイルのコンテンツを渡します。 ■ ffreturn という名前の入力ストリームまたは ByteArray でデータを返します。

Integration Server が使用するコンテンツハンドラ	コンテンツタイプ	解析する対象
		このコンテンツタイプヘッダーの使用方法の詳細については、『*Flat File Schema Developer's Guide*』を参照してください。
ContentHandler_IDAT	application/x-wmidatabin	HTTP 要求からの入力ストリームまたは FTP 要求からのドキュメント。 このコンテンツハンドラは、以下の処理を実行します。 ■ 入力ストリームまたはドキュメントを iData オブジェクトに変換します。 ■ データを IData の XML 表現に変換します。
ContentHandler_JSON	application/json	JSON コンテンツ。 **注:** Integration Server では、application/json コンテンツタイプは admin、invoke、rest、および restv2 ディレクティブでのみサポートされます。 このコンテンツハンドラの動作は、watt.server.http.jsonFormat パラメータの設定に基づいています。 ■ 「`parsed`」を設定した場合、JSON コンテンツをパイプライン変数に自動的に解析します。次に JSON を使用してパイプラインを構成し、pub.json:jsonStreamToDocument を呼び出すことなく、既存のサービスを実行できます。 ■ 「`stream`」を設定した場合、要求の本文を jsonStream としてパイプラインに追加します。フローサービスで jsonStream からの要素を使用できるように、**pub.json:jsonStreamToDocument** サービスは、jsonStream をドキュ

Integration Server が使用するコンテンツハンドラ	コンテンツタイプ	解析する対象
		メント (IData オブジェクト) に変換します。 ■ 「bytes」を設定した場合、JSON コンテンツの受信ストリームを jsonBytes というバイト配列として渡します。 **注:** watt.server.http.jsonFormat パラメータの詳細については、1095ページの「サーバ設定パラメータ」を参照してください。 pub.json:jsonStreamToDocument サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。
ContentHandler_Multipart	multipart/related multipart/mixed multipart/form-data	関連する本文パートの一部であるオブジェクト。たとえば、HTML Web ページとその関連する ../_graphics/is_administrator\s_guide_graphics/ graphics ファイル。 このコンテンツハンドラは、以下の処理を実行します。 ■ 入力ストリームをキー contentStream でパイプラインに渡します。 ■ キー/値のペアを HTML の表に書き込みます。
ContentHandler_RPC	application/x-wmrpc	HTTP 要求からの入力ストリームまたは FTP 要求からのドキュメント。 このコンテンツハンドラは、以下の処理を実行します。 ■ 入力ストリームからキー/値のペアへデータを変換します。

Integration Server が使用するコンテンツハンドラ	コンテンツタイプ	解析する対象
		■ RPC としてエンコードされた値をデータストリームに書き込みます。
ContentHandler_RPC2	application/x-wmrpc2	HTTP 要求からの入力ストリームまたは FTP 要求からのドキュメント。 このコンテンツハンドラは、以下の処理を実行します。 ■ 入力ストリームまたはドキュメントから iData オブジェクトを作成します。 ■ RPC としてエンコードされた値をデータストリームに書き込みます。
ContentHandler_SOAP	text/xml	HTTP 要求からの入力ストリームまたは FTP 要求からのドキュメント。 このコンテンツハンドラは、以下の処理を実行します。 ■ SOAP 要求メッセージをキー soapRequestData でパイプラインに追加します。 ■ SOAP 応答をキー soapResponseData でパイプラインに追加します。
	application/soap	SOAP メッセージ。
	application/soap+xml	XML としてシリアライズされた SOAP 1.2 メッセージ。
ContentHandler_XML	text/xml	XML コンテンツ **注:** Integration Server では、text/xml コンテンツタイプは invoke、rest、および restv2 ディレクティブでのみサポートされます。 このコンテンツハンドラの動作は、`watt.server.coder.responseAsXML` パラメータの設定に基づいています。

Integration Server が使用するコンテンツハンドラ	コンテンツタイプ	解析する対象
		この動作は REST V2 要求に対してのみ適用されます。 ■ [true] を設定した場合、Integration Server は適切な XML フォーマットで REST API 要求に対する XML 応答を受信します。 ■ [false] を設定した場合、Integration Server は、IData オブジェクトを含む変換された XML フォーマットで REST API 要求に対する XML 応答を受信します。
	application/xml	XML コンテンツ **注:** Integration Server では、application/xml コンテンツタイプは invoke、rest、および restv2 ディレクティブでのみサポートされます。 このコンテンツハンドラの動作は、watt.server.coder.responseAsXML パラメータの設定に基づいています。この動作は REST 要求に対してのみ適用されます。 ■ [true] を設定した場合、Integration Server は適切な XML フォーマットで REST API 要求に対する XML 応答を受信します。 ■ [false] を設定した場合、Integration Server は、IData オブジェクトを含む変換された XML フォーマットで REST API 要求に対する XML 応答を受信します。

メモ:

- 他の webMethods 製品および顧客アプリケーションは、これらのコンテンツタイプヘッダーについて異なるコンテンツハンドラを登録できます。それにより、Integration Server がそのタイプのメッセージを受信したときに Integration Server の動作が変わる場合があります。たとえば、Trading Networks は text/xml について固有の XML コンテンツハンドラを登録し

ます。この場合、Integration Server は Trading Networks のコンテンツハンドラを使用して受信 XML メッセージを処理します。

- コンテンツタイプヘッダーフィールドで、コンテンツハンドラが登録されていないコンテンツタイプが指定されている場合、Integration Server はコンテンツを text/html として処理するデフォルトのコンテンツハンドラ (ContentHandler_Default) を使用します。
- FTP または FTPS ポートが拡張子がないファイルを受信した場合、Integration Server はコンテンツを text/html として処理するデフォルトのコンテンツハンドラ (ContentHandler_Default) を使用します。
- Integration Server は、マルチパートコンテンツハンドラの関連コーダーと共に独自のサービスを登録できるようにする次の Java API を備えています。`com.wm.app.b2b.server.ServerAPI.registerCoderForMultipart(String serviceName, Class coder)`。次の Java API を使用して、コーダーの登録を解除できます。`com.wm.app.b2b.server.ServerAPI.removeCoderForMultipart(String serviceName)`。コーダーを登録していない場合、または特定のサービスの既存のコーダーを登録解除した場合、Integration Server はサービスのデフォルトのコンテンツハンドラ (ContentHandler_Default) を使用します。
- application/xml のコンテンツタイプを指定する Web サービス要求は、バージョン 9.12 より前の Integration Server のバージョンでは成功します。しかし、Integration Server バージョン 9.12 以降では、この Web サービス要求は失敗します。これは、Integration Server 9.12 より前のバージョンで、Integration Server はコンテンツタイプ application/xml の直接処理をあらかじめサポートしていなかったためです。application/xml に対するコンテンツハンドラが指定されていなかった場合、Integration Server ではデフォルトのコンテンツハンドラが使用されました。デフォルト処理は text/html の処理と同じです。XML は inputStream としてパイプラインに置かれます。バージョン 9.12 以降、Integration Server にはコンテンツタイプ application/xml をサポートする ContentHandler_XML という名前のコンテンツハンドラが追加されました。コンテンツは XML ノードとしてパイプラインに置かれます。この動作変更のため、Web サービス要求はバージョン 9.12 で失敗します。Web サービスでは XML ノードではなく inputStream を想定するためです。この問題に対処し、サービスが XML ノードではなく inputStream を受け付けるように変更する必要をなくすには、watt.server.content.type.mappings サーバ設定パラメータを使用して、application/xml コンテンツタイプを text/xml コンテンツタイプにマップできます。Integration Server では、application/xml のコンテンツタイプを指定する要求を受信したときに text/xml コンテンツハンドラを使用します。以下を指定します。

```
watt.server.content.type.mappings=application/xml text/xml
```

HTTP 応答のコンテンツハンドラについて

Integration Server は以下の処理を行って、HTTP 応答にコンテンツハンドラを選択します。

1. 応答ヘッダーのコンテンツタイプヘッダーを確認して、そのコンテンツタイプ用に登録されているコンテンツハンドラを使用します。このフィールドには、アプリケーションによって pub.flow:setResponseHeader サービスで値が割り当てられている場合にのみ、値が含まれます。

応答ヘッダーのコンテンツタイプヘッダーフィールドが、サポートされていないコンテンツタイプを指定している場合、Integration Server は text/html を使用します。

2. コンテンツタイプが応答ヘッダーで指定されていない場合、Integration Server は要求ヘッダーの承認ヘッダーフィールドを確認してコンテンツタイプを決定し、そのコンテンツタイプ用に登録されているコンテンツハンドラを使用します。
3. 要求ヘッダーに承認ヘッダーフィールドがない場合、または承認ヘッダーフィールドで指定されているコンテンツタイプのコンテンツハンドラがない場合、Integration Server は要求で使用したコンテンツハンドラを使用します。

注:
応答が既に pub.flow:setResponse2 サービスまたは pub.flow:HTTPResponse ドキュメントタイプによって作成されている場合、Integration Server は応答用のコンテンツハンドラを使用しません。

承認ヘッダーフィールド

承認ヘッダーフィールドには、クライアントが応答で受け入れることになる 1 つまたは複数のコンテンツタイプが指定されています。承認ヘッダーフィールドでは、コンテンツタイプをリストするか (application/json、text/html、text/xml など) またはワイルドカードを使用することで (text/*)、受け入れ可能な複数のコンテンツタイプを指定できます。

デフォルトでは、承認ヘッダーでワイルドカードを使用してコンテンツタイプを指定している場合、Integration Server は、その主要タイプ用 (text、application、multipart) に登録されている最初のコンテンツハンドラを選択します。

次の表に、承認ヘッダーフィールドの値に応じて Integration Server が選択するコンテンツハンドラを示します。

承認ヘッダーフィールド	選択されるコンテンツハンドラ
text/*	XML
application/*	CGI
multipart/*	Multipart

承認ヘッダーでワイルドカードが指定されている場合に Integration Server によって使用されるコンテンツハンドラを制御するには、watt.server.content.type.mappings サーバ設定パラメータを使用する必要があります。このパラメータの構文は次のとおりです。

watt.server.content.type.mappings=<*wildcard1*> <*content-type1*>,<*wildcard2*> <*content-type2*>,<*wildcardN*> <*content-typeN*>

たとえば、`text/*` を `text/xml`、`multipart/*` を `multipart/related` に関連付けるには、以下のようにパラメータを指定します。

watt.server.content.type.mappings=text/* text/xml,multipart/* multipart/related

承認ヘッダーには、優先順序を示す複数のコンテンツタイプとパラメータを含めることができます。Integration Server は、RFC 2616 の第 14 項で定義されている最も優先順序の高いコンテンツタイプを使用して応答を試行します。

要求では、承認ヘッダーに複数のメディアタイプが含まれるが、いずれの係数の重み (q パラメータ) も指定されていない場合、Integration Server は承認ヘッダーの最初のメディアタイプで応答を試みます。最初のメディアタイプにコンテンツハンドラが定義されていない場合、承認ヘッダーの 2 番目のメディアタイプで応答を試みます。Integration Server が承認ヘッダー内で応答できるメディアタイプを見つけるまで、この処理は繰り返されます。何も見つからなかった場合、つまり、承認ヘッダー内のメディアタイプに対して登録されているコンテンツハンドラがない場合、Integration Server は text/html メディアタイプで応答します。

Software AG では、メディアタイプが複数ある承認ヘッダーをクライアントが送信する場合に応答のメディアタイプの優先順序を示す q パラメータを使用することをお勧めします。

watt.server.http.useAcceptHeader 設定パラメータが「`true`」に設定されている場合、Integration Server によって HTTP 要求ヘッダーの承認ヘッダーフィールドがサポートされます。このパラメータのデフォルト設定は、「`true`」です。

FTP 応答のコンテンツハンドラについて

応答を作成するとき、Integration Server は、lib/mime.types ファイルでコンテンツタイプ用に登録されたコンテンツハンドラを使用します。

Integration Server_directory /instances/*instance_name*/lib/mime.types ファイルには、ファイル拡張子とコンテンツタイプのマッピングが保存されています。[**設定**] > [**リソース**] > [**MIME タイプ**] を選択することで、Integration Server Administrator でマッピングを編集できます。lib/mime.types ファイルのマッピングを編集する方法の詳細については、『*webMethods Service Development Help*』を参照してください。

注:
応答が既に pub.flow:setResponse2 サービスによって作成されている場合、Integration Server は応答用のコンテンツハンドラを使用しません。

52 Integration Server におけるクラスロード

- Integration Server におけるクラスロードの動作 964
- クラスをサーバクラスパスに追加する 968
- クラスファイルと jar ファイルの配置場所 974
- クラスロードの迅速化 975

Integration Server におけるクラスロードの動作

Integration Server は JVM (Java Virtual Machine : Java 仮想マシン) 上で実行されます。Integration Server、または任意の Java アプリケーションでクラスへのアクセスが必要になると、JVM はそのクラスの実行可能なバイトコードを探して、JVM に取り込みます。このプロセスは「クラスのロード」と呼ばれ、クラスローダによって処理されます。 Integration Server のクラスローダおよび JVM はさまざまな場所からクラスをロードするよう設計されています。

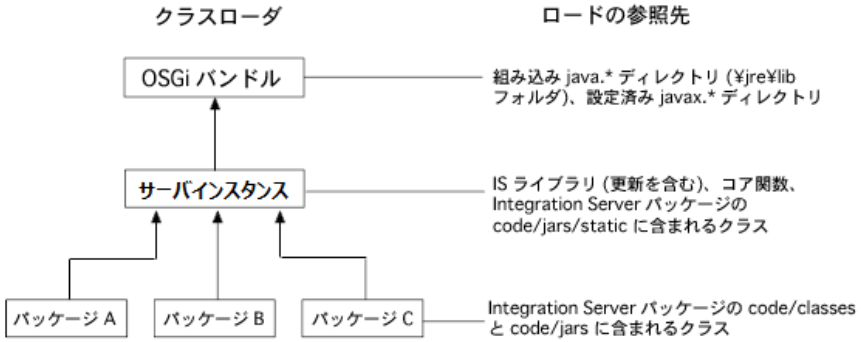
Integration Server では、サーバの初期化中、パッケージのロード時に、Java サービスを含むすべてのクラスも Integration Server によってロードされます。Java サービスには、他の Java クラスを参照するステートメントが含まれます。これらのステートメントを実行するには、参照先クラスを JVM にロードする必要があります。

自社で開発した独自のクラスまたはサードパーティライブラリのクラスを使用する場合、そのクラスの可視性とロード順序を制御できます。クラスの可視性とは、クラスと jar ファイルが Integration Server のすべてのクラスで見えるようにする (またはその必要がある) か、特定のパッケージでのみ見えるようにする (またはその必要がある) かを示します。

クラスローダ

Integration Server は Eclipse Equinox でサポートされている OSGi プラットフォーム上で実行されます。Integration Server の親クラスローダは OSGi バンドルクラスローダです。

次の図は、クラスローダチェーンにおけるそれぞれのクラスローダとその関係を示しています。



注:
Microservices Runtime では OSGi を使用しません。Microservices Runtime の親クラスローダは Java システムクラスローダです。

OSGi バンドルクラスローダ

OSGi バンドルクラスローダは、Integration Server の親のクラスローダです。OSGi バンドルクラスローダは、OSGi フレームワーク (Eclipse Equinox) によって提供されるものであり、Integration Server に組み込まれています。

注:
Microservices Runtime の親クラスローダは Java システムクラスローダです。

デフォルトでは、OSGi バンドルクラスローダは、CLASSPATH 環境変数または一部の Java 専用パラメータ (-javaagent パラメータなど) で指定された jar ファイルにアクセスすることができません。CLASSPATH 環境変数または Java 専用パラメータで指定された jar ファイルにアクセスする場合は、*Software AG_directory*\profiles\IS_*instance_name*\config.ini ファイルに以下のエントリを追加する必要があります。

```
osgi.parentClassloader=app
```

注:
osgi.parentClassloader プロパティを config.ini ファイルに追加した後は、Integration Server を再起動する必要があります。

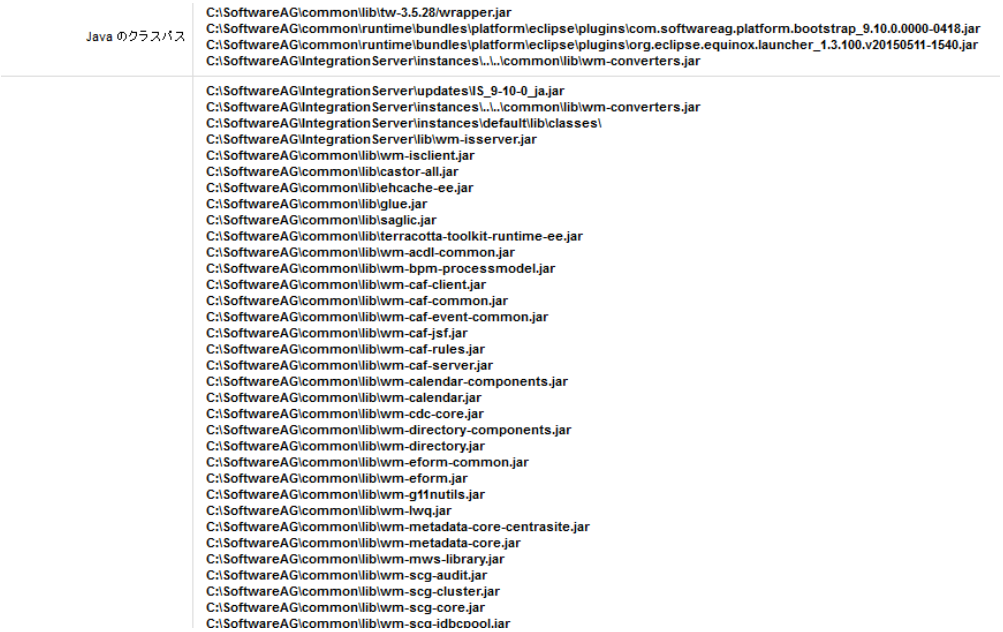
Integration Server クラスローダ

Integration Server にはサーバクラスローダとパッケージクラスローダが含まれます。これらのクラスローダでは、Integration Server が機能するために必要なクラスをロードします。

- Integration Server のサーバクラスローダは、Integration Server のコアを構成するクラスをロードします。このローダは**サーバのクラスパス**からロードします。サーバのクラスパスは、この後で説明する [製品情報] ページで確認できます。
- Integration Server のパッケージクラスローダは、Integration Server パッケージからクラスをロードします。これらのパッケージには、Designer を使用してユーザが作成したパッケージや Integration Server の一部として提供されている事前定義済みのパッケージがあります。パッケージにはそれぞれ専用のクラスローダがあります。

クラスパス

Integration Server クラスパスは、Integration Server クラスローダがクラスを探すときに検索するファイルとディレクトリのリストです。Integration Server Administrator の [**製品情報**] ページには、Integration Server のクラスローダによって使用されるクラスパスを構成するファイルとディレクトリが表示されます。次の図は、[製品情報] 画面の一部を示しています。



Integration Server クラスパスの設定方法

Integration Server を起動すると、startup.bat (Windows の場合) または startup.sh (UNIX の場合) という名前のスクリプトファイルが実行されます。startup.bat/sh ファイルは、多数のディレクトリおよび変数で構成される Integration Server のクラスパス変数を構築します。

次の表に、Integration Server が Integration Server クラスパスを構築するために使用するディレクトリおよび変数を示します。

ディレクトリまたは変数	説明
Software AG_directory\profiles\IS_*instance_name*\configuration\custom_wrapper.conf の `wrapper.java.additional.202=-Dwatt.server.prepend.classes` プロパティ	custom_wrapper.conf で定義されている変数。クラスパスの先頭に追加するディレクトリとファイルを定義します。
Software AG_directory\profiles\IS_*instance_name*\configuration\custom_wrapper.conf の `wrapper.java.additional.203=-Dwatt.server.append.classes` プロパティ	custom_wrapper.conf で定義されている変数。クラスパスの末尾に追加するディレクトリとファイルを定義します。
*Software AG_directory**Integration Server_directory*\instances*instance_name*\bin\ini.cnf	*Integration Server_directory*\instances*instance_name*\bin\ini.cnf の application.classpath プロパティ。通常、次のディレクトリのクラスがこの順序で指定されます。 ■ *Integration Server_directory*\lib

ディレクトリまたは変数	説明
	■ *Software AG_directory* \common\lib
*Software AG_directory**Integration Server_directory*\instances*instance_name*\lib\jars\custom	このディレクトリにあるすべてのカスタム/サードパーティの .jar および .zip ファイル。 **注:** このディレクトリを使用して、サーバの特定インスタンスに対して使用可能にするカスタム/サードパーティの .jar および .zip ファイルを保存します。
*Software AG_directory**Integration Server_directory*\instances*instance_name*\packages*package_name*\code\jars\static	各パッケージのこのディレクトリにあるすべての .jar および .zip ファイル。ファイルは、File.list() メソッドで返された順序でクラスパスに追加されます。 **注:** このディレクトリにある Jar ファイルは、関連するパッケージが無効になっている場合でもロードされます。
*Software AG_directory**Integration Server_directory*\instances*instance_name*\packages*package_name*\code\classes	このディレクトリが存在する場合、クラスパスに追加されます。
*Software AG_directory**Integration Server_directory*\instances*instance_name*\packages*package_name*\code\classes.zip	このファイルが存在する場合、クラスパスに追加されます。
*Software AG_directory**Integration Server_directory*\instances*instance_name*\updates	正当かつ無効にされていない更新および修正。
*Software AG_directory**Integration Server_directory*\lib\jars	このディレクトリにあるすべての .jar および .zip ファイル。
*Software AG_directory**Integration Server_directory*\lib\jars\custom	このディレクトリにある共通のカスタム/サードパーティの .jar および .zip ファイル。 **注:** このディレクトリを使用して、サーバのすべてのインスタンスに対して使用可能にするカスタム/サードパーティの .jar および .zip ファイルを保存します。

ディレクトリまたは変数	説明
*Software AG_directory**Integration Server_directory*\updates	共通で、正当かつ無効にされていない更新および修正。 **注:** このディレクトリ内の更新および修正は、すべてのサーバインスタンスが利用できます。

注:
場合によっては、Integration Server Administrator の [製品情報] ページに、存在しないファイルの名前が表示されることがあります。Integration Server クラスパスの `wrapper.java.additional.202=-Dwatt.server.prepend.classes` および `wrapper.java.additional.203=-Dwatt.server.append.classes` プロパティに指定されているファイル名は、存在しなくても Integration Server の [製品情報] ページに表示されます。ini.cnf ファイルが参照している jar ファイルが存在しない場合も、[製品情報] ページに表示されます。watt.debug.warnOnClasspathError サーバ設定パラメータを「true」に設定した場合、以下のエラーが存在しない jar ファイルごとに標準出力に書き込まれます。**`ini.cnf` のクラスパスエントリが見つかりません:** `<jar_file_name>`。標準出力は *Software AG_directory*/profiles/IS_*instance_name*/logs/wrapper.log に送信されます。

Microservices Runtime クラスパスの設定方法の詳細

Microservices Runtime クラスパスは Integration Server クラスパスとほとんど同じように設定されます。ただし、2 つの大きな違いがあります。

- Microservices Runtime ではインスタンスを使用しません。966 ページの「Integration Server クラスパスの設定方法」 の表における Integration Server の *Software AG_directory**Integration Server_directory*\instances*instance_name* への参照は Microservices Runtime の *Integration Server_directory* に対応します。
- Microservices Runtime では OSGi を使用しません。したがって、custom_wrapper.conf ファイルの情報を使用してクラスパスを構築しません。代わりに、Microservices Runtime は *Integration Server_directory*/bin/setenv.bat(sh) の情報 (具体的には PREPENDCLASSES、APPENDCLASSES の各プロパティの情報) を使用します。

注:
場合によっては、Microservices Runtime の [製品情報] ページに、存在しないファイルの名前が表示されることがあります。具体的には、Microservices Runtime クラスパスの PREPENDCLASSES および APPENDCLASSES 変数に指定されているファイル名は、存在しなくても Microservices Runtime の [製品情報] ページに表示されます。

クラスをサーバクラスパスに追加する

自社で開発した独自のクラスまたはサードパーティライブラリのクラスを使用する場合、そのクラスの可視性とロード順序を制御できます。クラスの可視性とは、クラスと jar ファイルが

Integration Server のすべてのクラスで見えるようにする (またはその必要がある) か、特定のパッケージでのみ見えるようにする (またはその必要がある) かを示します。

サーバクラスパスに追加されたクラスと jar ファイルは、すべての Java サービスを含む Integration Server のすべてのクラスで見ることができます。ある意味で、クラスと jar ファイルの可視性はグローバルです。

注:
クラス宣言から「public」修飾子を省略すると、クラスの可視性は同じ Java パッケージのメンバーに制限できます。このことはクラスのロードに直接関係なく、Java のマニュアルに記述されています。

クラスとライブラリ (jar ファイルと zip ファイル) をクラスパスに追加するには、以下の 3 つの方法があります。

- **オプション 1**jar ファイルまたは zip ファイルをパッケージの code\jars\static ディレクトリにコピーします。Integration Server では、パッケージのロード時にクラスパスにファイルを追加します。

  パッケージの code\jars\static jar ディレクトリにある jar ファイルの内容を変更した場合、変更内容を有効にするには、Integration Server を再起動する必要があることに注意してください。パッケージを再ロードするだけでは十分でありません。

- **オプション 2***Software AG_directory**Integration Server_directory*\instances*instanceName*\lib\jars\custom ディレクトリに jar ファイルまたは zip ファイルをコピーします。Integration Server では、Integration Server の起動時にクラスパスにファイルを追加します。

  インストールされたすべての Integration Server インスタンスのクラスパスに jar ファイルまたは zip ファイルを追加する場合、*Software AG_directory**Integration Server_directory*\lib\jars\custom ディレクトリにファイルをコピーします。

  注:
  Microservices Runtime に対するオプション 2 の場合、jar ファイルまたは zip ファイルを *Software AG_directory**Integration Server_directory*\lib\jars\custom に追加します。

- **オプション 3**ファイルシステムの別の場所にファイルをコピーします。jar ファイルまたはクラスファイルディレクトリへのパスを次のファイルの watt.server.prepend.classes または watt.server.append.classes プロパティセットに追加します。

  Software AG_directory\profiles\IS_*instanceName*\configuration\custom_wrapper.conf

  custom_wrapper.conf ファイルのクラスパス情報が上書きされている場合、サーバは起動時にそれらの設定を使用します。

  Integration Server クラスパスの先頭および末尾にディレクトリを追加するのに使用できる変数は、以下の表のとおりです。

ディレクトリの追加先	編集する変数
クラスパスの先頭	`wrapper.java.additional.202=` `-Dwatt.server.prepend.classes=`

ディレクトリの追加先	編集する変数
クラスパスの末尾	`wrapper.java.additional.203=` `-Dwatt.server.append.classes=`

Microservices Runtime の場合、jar ファイルまたはクラスファイルへのパスを *Integration Server_directory*/bin/setenv.bat(sh) に追加します。

次の表は、Microservices Runtime のクラスパスの先頭または末尾にディレクトリを追加するため、setenv.bat(sh) に設定できる Java システムプロパティを示します。

ディレクトリの追加先	編集対象の Java システムプロパティ
クラスパスの先頭	setenv.bat/sh の PREPENDCLASSES
クラスパスの末尾	setenv.bat/sh の APPENDCLASSES

これらの変数をクラスパスに追加する方法と追加する場所の詳細については、965 ページの「クラスパス」を参照してください。custom_wrapper.conf と設定可能なプロパティの詳細については、*Software AG Infrastructure Administrator's Guide*を参照してください。

クラスとライブラリをクラスパスに追加する方法を選択する場合は、以下の情報に留意してください。

- 上記のオプション 1 と 2 は jar ファイルと zip ファイルにのみ有効です。オプション 3 は jar ファイル、zip ファイル、クラスファイル、およびその他のファイルに有効です。
- 一部のサードパーティライブラリは、クラスパス内で設定ファイルまたは他の種類のファイルを調べます。これらのファイルをクラスパスに追加するには、オプション 3 を使用します。
- オプション 1 を使用した場合、パッケージが別の Integration Server に展開されると、パッケージの code\jars\static ディレクトリにあるすべての jar ファイルと zip ファイルは自動的に含められます。オプション 2 と 3 の場合、jar ファイル、zip ファイルまたはクラスファイルを展開するには、追加手順が必要です。
- サーバのクラスパス上でクラスが置かれる位置が、ロード順序に反映されます。ロード順序の詳細については、972 ページの「クラスの検索方法」を参照してください。

パッケージへのクラスの追加

クラスが単一の Integration Server パッケージのサービスでのみ使用される場合、パッケージの code/classes ディレクトリにクラスを配置することで、クラスの可視性をそのパッケージに制限することができます。クラスを jar ファイルまたは zip ファイルで配布する場合、パッケージの code/jars ディレクトリにそのファイルを入れます。パッケージの code/classes または code/jars ディレクトリにあるファイルは Integration Server のクラスパスに追加されません。

注:

パッケージの code/jars または code/classes ディレクトリの jar ファイルまたはクラスファイルの内容を変更した場合、パッケージを再ロードして変更内容を有効にします。変更した内容を有効にするために Integration Server を再起動する必要はありません。

別のパッケージの code/classes または code/jars ディレクトリにあるクラスへのアクセスが必要なパッケージがある場合、2つのパッケージ間の依存関係を作成することでアクセス可能にできます。たとえば、次のシナリオについて考えます。

- PackageA の code/classes ディレクトリにはクラスがあります。
- PackageB では、PackageA の code/classes ディレクトリのクラスを使用する必要があります。

Designer を使用して、PackageB から PackageA への依存性を確立できます。

または、PackageA を必要とするよう PackageB の manifest.v3 を変更できます。

```
<record name="requires" javaclass="com.wm.util.Values">

<value name="PackageA">1.0</value>

</record>
```

マニフェストの編集後、Integration Server が実行中であれば、PackageB を再ロードします。

PackageB は、PackageA の code/classes および code/jars ディレクトリのクラスにアクセスできるようになります。PackageB は PackageA を必要とするため、PackageA が有効にならない限り、PackageB は有効にできません。PackageB を無効にしない限り、PackageA は無効にできません。PackageA を再ロードすると、PackageB が再ロードされます。

クラスの検索順序

Java サービスでクラスを参照する場合、Integration Server ではデフォルトで以下の順序で参照先クラスを検索します。

1. サーバのクラスパス
2. ローカルのパッケージ内
3. すべての必須パッケージ内

クラスが見つかると、Integration Server はすぐにクラスをロードして使用します。

クラスのロード順序を変更するには、パッケージのクラスローダを使用するようパッケージの manifest.v3 ファイルを変更します。

```
<value name='classloader'>package</value>
```

パッケージの `classloader` プロパティが「package」である場合、クラスのロード順序は以下のとおりです。

1. ローカルのパッケージ内
2. すべての必須パッケージ内

3. サーバのクラスパス

`classloader` プロパティのデフォルト値は「system」です。classloader プロパティに値が設定されていない場合、Integration Server はこの値を使用します。

```
<value name='classloader'/>
```

パッケージのクラスローダを使用するようパッケージを設定する方法の詳細については、742 ページの「パッケージクラスローダの使用」を参照してください。

クラスの検索方法

968 ページの「クラスをサーバクラスパスに追加する」で説明したように、サーバのクラスパスでクラスを置く位置を選択する場合、位置によるロード順序への影響を考慮します。Integration Server では、さまざまな場所を以下の順序で検索します。

1. watt.server.prepend.classes
2. *Software AG_directory**Integration Server_directory*\instances*instanceName*\bin\ini.cnf

   具体的には、ini.cnf ファイルの application.classpath プロパティです。通常、application.classpath プロパティは次のディレクトリのクラスをこの順序で指定します。

   - *Integration Server_directory*\lib
   - *Software AG_directory*\common\lib
3. *Software AG_directory**Integration Server_directory*\instances*instanceName*\lib\jars
4. *Software AG_directory**Integration Server_directory*\instances*instanceName*\lib\jars\custom
5. *Software AG_directory**Integration Server_directory*\lib\jars
6. *Software AG_directory**Integration Server_directory*\lib\jars\custom
7. *Software AG_directory**Integration Server_directory*\instances*instanceName*\packages*packageName*\code\jars\static
8. watt.server.append.classes

注:
Microservices Runtime がクラスファイルと jar ファイルを検索するディレクトリの詳細については、973 ページの「Microservices Runtime のクラスの検索方法」を参照してください。

クラスが単一のクラスファイルか jar ファイル内のクラスかは関係なく、クラスのコピーが複数あり、サーバのクラスパス内にクラスの複数の場所がある場合、Integration Server はサーバのクラスパスで最初に置かれたクラスのインスタンスを常にロードします。Integration Server はサーバのクラスパスで後に置かれた同じクラスの他のインスタンスを使用しません。このことが重要になるのは、サードパーティの jar ファイルの異なるバージョンがサーバのクラスパスにある場合です。Integration Server はサーバのクラスパスで最初に置かれたバージョンを常に使用します。

複数のパッケージの code\jars\static ディレクトリに jar ファイルがある場合、jar ファイルはパッケージ名のアルファベット順にサーバのクラスパスに追加されます。たとえば、PackageA\code\jars\static の jar ファイルは PackageB\code\jars\static の jar ファイルの前にロードされます。

Integration Server のクラスパスは Integration Server Administrator の [製品情報] ページで確認できます。

Microservices Runtime のクラスの検索方法

Microservices Runtime が検索する場所の順序は Integration Server の場合と異なります。これは、Microservices Runtime では OSGi を使用せず、\instances*instanceName* ディレクトリがないためです。

Microservices Runtime では、さまざまな場所を以下の順序で検索します。

1. *Integration Server_directory*\bin\server.bat(sh) の PREPENDCLASSES
2. *Software AG_directory**Integration Server_directory*\bin\ini.cnf

   具体的には、ini.cnf ファイルの application.classpath プロパティです。通常、application.classpath プロパティは次のディレクトリのクラスをこの順序で指定します。

   - *Integration Server_directory*\lib
   - *Software AG_directory*\common\lib
3. *Software AG_directory**Integration Server_directory*\lib\jars
4. *Software AG_directory**Integration Server_directory*\lib\jars\custom
5. *Software AG_directory**Integration Server_directory*\packages*packageName*\code\jars\static
6. *Integration Server_directory*\bin\server.bat(sh) の APPENDCLASSES

注:
Integration Server と同じように、Microservices Runtime は code\jars\static ディレクトリの jar ファイルで 2 つ以上のクラスのコピーまたは複数のパッケージを処理します。詳細については、972 ページの「クラスの検索方法」を参照してください。

パッケージ内の検索順序

Integration Server は、次の順序を使用して、パッケージ内でクラスを検索します。

1. *packageName*\resources
2. *packageName*\lib
3. *packageName*\code\jars
4. *packageName*\code\classes

注:

パッケージに別のパッケージが必要な場合、Integration Server はクラスを参照するパッケージ内で上記の順序で参照先クラスを検索します。次に、Integration Server は必要なパッケージ内で上記の順序で検索します。

クラスファイルと jar ファイルの配置場所

自社で開発した独自のクラスまたはサードパーティライブラリのクラスを使用する場合、そのクラスの可視性とロード順序を制御できます。クラスの可視性とは、クラスと jar ファイルが Integration Server のすべてのクラスで見えるようにする (またはその必要がある) か、特定のパッケージでのみ見えるようにする (またはその必要がある) かを示します。

クラスファイルと jar ファイルの配置場所を決定するには、以下のガイドラインが役立ちます。

- クラスファイルまたは jar ファイルが 1 つの Integration Server パッケージのサービスでのみ必要とされる場合、パッケージの code/classes または code/jars ディレクトリに入れます。
- クラスファイルまたは jar ファイルが 2 番目の Integration Server パッケージで必要とされる場合、クラスまたは jar はすでにあるパッケージ内に保持し、最初のパッケージにパッケージの依存性を設定します。それには、Designer を使用して 2 番目のパッケージから最初のパッケージへの依存性を設定するか、最初のパッケージを必要とするよう 2 番目のパッケージの manifest.v3 ファイルを変更します。
- さらに他の Integration Server パッケージでクラスファイルまたは jar ファイルが必要であり、パッケージの依存性の管理が難しくなった場合、以下のいずれかの方法を実行します。
  - jar ファイルまたは zip ファイルの場合、パッケージの code/jars/static ディレクトリにそのファイルを移動します。jar または zip はサーバのクラスパスに追加され、すべてのパッケージで利用できるようになります。
  - 1 つまたは複数のクラスであり、Integration Server を使用している場合、次のファイルのいずれか 1 つの行に親ディレクトリへのフルパスを追加します。*Software AG_directory*\profiles\IS_*instanceName*\configuration\custom_wrapper.conf

    ```
    wrapper.java.additional.202=-Dwatt.server.prepend.classes=

    wrapper.java.additional.203=-Dwatt.server.append.classes=
    ```

- 1 つまたは複数のクラスであり、Microservices Runtime を使用している場合、次のファイルのいずれか 1 つの Java システムプロパティに親ディレクトリへのフルパスを追加します。*Software AG_directory**Integration Server_directory*\bin\setenv.bat(sh)

  APPENDCLASSES

  PREPENDCLASSES
- 別のパッケージにあるバージョンとは異なるバージョンのクラスファイルまたは jar ファイルがお使いのパッケージで必要な場合、お使いのパッケージの code/classes または code/jars ディレクトリに必要なバージョンをコピーします。
- サーバのクラスパスにあるバージョンとは異なるバージョンのクラスファイルまたは jar ファイルがパッケージで必要な場合、以下の手順に従います。

1. パッケージの code/classes または code/jars ディレクトリに必要なバージョンをコピーします。
2. パッケージの classloader プロパティを「package」に変更します。

- Java セキュリティプロバイダが JVM で登録されている場合、このプロバイダはすべての Integration Server パッケージで利用できます。Java セキュリティの実装では、JVM がプロバイダの実装をそのクラスローダと共に保持し、JVM 内で実行されているすべてのプロセスでこの実装が利用できるようになっています。このことは、他の Integration Server パッケージ、および JVM で実行される他のインストール済み製品にも影響を与えます。そのため、追加セキュリティプロバイダの登録や既存のセキュリティプロバイダの上書きは慎重に行う必要があります。

クラスロードの迅速化

クラスロードは多数のディレクトリおよびクラスが関わる場合には、時間を消費する処理になる可能性があります。アプリケーションおよびその使用方法によっては、以下のサーバ設定パラメータでクラスロードをより迅速にすることができます。

watt.server.coder.bincoder.trycontextloaderfirst

このプロパティを「true」に設定すると、Integration Server がパイプラインをエンコードまたはデコードするときに、コンテキストローダを使用してから、現在実行しているスレッドのクラスローダを使用するようになります。参照されるクラスが特定のパッケージに属している場合、最初にコンテキストローダを使用した方が処理が速くなることがあります。

watt.server.classloader.pkgpriority=packageName, packageName

一部の Integration Server 機能はパッケージのコンテキスト内で動作しません。こうした機能からクラスが参照されると、特に決まった順序なしにすべてのパッケージが検索されます。この設定パラメータでは、Integration Server が最初に検索するパッケージのカンマ区切りリストを指定します。パッケージの検索順序を指定すると、パフォーマンスが向上することがあります。このプロパティの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

53 Integration Server におけるホワイトリストフィルタ処理

- Integration Server におけるホワイトリストフィルタ処理について 978
- ホワイトリストクラスフィルタ処理の有効化または無効化 978
- Integration Server ホワイトリストクラスファイル 978
- パッケージ用ホワイトリストクラスファイルの作成 979
- ホワイトリストにないクラスをログに記録する 980
- Integration Server ブラックリストクラスファイル 980

Integration Server におけるホワイトリストフィルタ処理について

Integration Server では、ホワイトリストフィルタを使用して、安全でない Java オブジェクトのデシリアライズを回避します。シリアライズされたオブジェクトを受け付けるが、信用のない入力を適切にデシリアライズする前に妥当性検査またはチェックを行わないアプリケーションには、Java デシリアライズの脆弱性が存在します。悪意のあるユーザはこの脆弱性を利用して悪意のあるオブジェクトをデータストリームに挿入し、リモートコード実行、サービス拒否などのさまざまな攻撃を行う可能性があります。

Integration Server では、IData の転送に使用される IDataBinCoder プロトコルがシリアライズされた Java オブジェクトをサポートしています。この機能が悪用されて、悪意のあるシリアライズされたペイロードの作成に使用できるサードパーティライブラリを Integration Server がロードするよう設定されることがあります。これにより、任意のコード実行や他の重大なセキュリティ上の問題を引き起こす可能性があります。この脆弱性に対処しないと悪用され、実稼働環境が破損することがあります。

Integration Server が信用のない Java オブジェクトをデシリアライズしないようにするには、Integration Server でロードしデシリアライズできるクラスのホワイトリストを Integration Server に含めます。ホワイトリストフィルタ処理を有効にすると、Integration Server では、クラスがホワイトリストにある場合にのみ、Java オブジェクトをデシリアライズします。Integration Server でクラスがホワイトリストにない Java オブジェクトが見つかると、Integration Server は ClassNotFoundException をスローします。

Integration Server には、サーバのホワイトリストクラスファイルが含まれます。また、個々のパッケージ用のホワイトリストクラスファイルを作成することもできます。このファイルは、パッケージで使用されるカスタムクラスの識別に使用できます。

ホワイトリストクラスフィルタ処理の有効化または無効化

Integration Server でホワイトリストフィルタ処理を実行するかどうかは、サーバ設定パラメータ watt.server.checkWhitelist で制御します。このパラメータを true に設定すると、ホワイトリストフィルタ処理は有効になります。Integration Server では、クラスがホワイトリストにある場合にのみ、Java オブジェクトをデシリアライズします。Integration Server でクラスがホワイトリストにない Java オブジェクトが見つかると、ClassNotFoundException がスローされます。ホワイトリストフィルタ処理を有効または無効にするには、Integration Server Administrator を使用して、watt.server.checkWhitelist パラメータの値を変更します。

watt.server.checkWhitelist 値に関係なく、Integration Server では Java オブジェクトのデシリアライズ時にブラックリストフィルタ処理を実行します。詳細については、以下を参照してください。 980 ページの「Integration Server ブラックリストクラスファイル」

Integration Server ホワイトリストクラスファイル

Software AG は Integration Server および階層化された製品で使用されるクラスを識別するホワイトリストを作成しました。次のファイルには、ホワイトリストクラスのリストが含まれます。

Integration Server_directory /instances/*instanceName*/config/security/whitelistclasses.xml

必要に応じて whitelistclasses.xml ファイルを編集し、カスタムクラスを含めることができます。

注:
Integration Server の修正または Integration Server の後続リリースで whitelistclasses.xml ファイルの内容が変更される可能性があります。その場合、whitelistclasses.xml ファイルのカスタマイズ内容は失われます。別の方法として、979 ページの「パッケージ用ホワイトリストクラスファイルの作成」で説明しているように、個々のパッケージ用のホワイトリストの作成と使用を検討してください。

Integration Server whiteslistclasses.xml ファイルを編集するには

1. 次のファイルをテキストエディタで開きます。

   Integration Server_directory /instances/*instanceName*/config/security/whitelistclasses.xml

2. classlist 配列に行「`<value>custom_class_name</value>`」を追加します。

3. ホワイトリストクラスファイルに追加するクラスごとに、前の手順を繰り返します。

4. 変更を保存してファイルを閉じます。

5. Integration Server を再起動します。

パッケージ用ホワイトリストクラスファイルの作成

Integration Server では、個々のパッケージに対するホワイトリストクラスのリストの作成と使用をサポートしています。Java オブジェクトをデシリアライズする Java サービスがカスタムパッケージに含まれている場合、役に立つことがあります。パッケージがロードされると、Integration Server はメモリに保持されているクラスの許可リストにパッケージのホワイトリストクラスを追加します。パッケージが無効になると、Integration Server はメモリに保持されているクラスの許可リストからパッケージのホワイトリストクラスを削除します。

注:
Integration Server がパッケージのホワイトリストクラスファイルを使用する場合、Integration Server のホワイトリストフィルタ処理を有効にしておく必要があります。watt.server.checkWhitelist parameter が「true」に設定されている場合、Integration Server はホワイトリストフィルタ処理を実行します。

パッケージホワイトリストを作成するには

1. 次の場所にある Integration Server whitliestclasses.xml ファイルをコピーします。

   Integration Server_directory /instances/*instanceName*/config/security/whitelistclasses.xml

2. コピーした whitelistclasses.xml ファイルを次の場所に貼り付けます。

   Integration Server_directory /instances/*instanceName*/packages/*packageName*/config

3. パッケージの whiteilstclasses.xml ファイルをテキストエディタで開きます。

4. すべての <value> 要素を削除します。

5. パッケージのホワイトリストクラスファイルに含めるカスタムクラスごとに 1 行を追加します。

   ```
   <value>custom_class_name</value>
   ```

6. パッケージの whiteilstclasses.xml ファイルの変更を保存します。

7. ホワイトリストを作成したパッケージを再ロードします。

ホワイトリストにないクラスをログに記録する

Integration Server を設定して、デシリアライズを試行したが whiteslistclasses.xml ファイルの一部でないクラスを示すログファイルを生成できます。ログファイルはデバッグに役立つことがあります。watt.server.dumpWhitelist が「true」に設定されている場合、Integration Server では、デシリアライズを試行したがホワイトリストクラスファイル内にないクラスを示すログファイルを生成します。具体的には、Integration Server はシャットダウン時に次のファイルを作成します。

Integration Server_directory /instances/*instanceName*/logs/runtimeClasslist.*time*.xml。*time* は 1970 年 1 月 1 日を起点とする時間です (ミリ秒単位)。

Integration Server が runtimeClasslist ファイルを生成するには、watt.server.checkWhitelist と watt.server.dumpWhitelist パラメータが「true」に設定され、Integration Server が最後に起動した以降にホワイトリストに含まれていないクラスのロードが試行されている必要があります。

Integration Server ブラックリストクラスファイル

Integration Server はオブジェクトのデシリアライズ時にブラックリストクラスフィルタ処理を実行します。Integration Server はオブジェクトのデシリアライズ時に安全でないクラスの存在を検出し、ClassNotFound 例外をスローします。そのため、そのクラスはインスタンス化されません。

Software AG で作成したブラックリストファイルが Integration Server に提供されています。ブラックリストは *Integration Server_directory* instances/*instanceName*/config/security/classlist.xml にあります。

ブラックリストクラスファイルはカスタマイズ可能です。

注:
Integration Server の修正または Integration Server の後続リリースでブラックリストファイル classlist.xml の内容が変更される可能性があります。その場合、classlist.xml ファイルのカスタマイズ内容は失われます。更新されたファイルに変更内容を再び追加する必要があります。

Integration Server ブラックリストクラスファイルを編集するには

1. テキストエディタを使用して、次のファイルを開きます。

   Integration Server_directory instances/*instanceName*/config/security/classlist.xml

2. classlist 配列に行「<value>*custom_class_name*</value>」を追加します。
3. ブラックリストクラスファイルに追加するクラスごとに、前の手順を繰り返します。
4. 変更を保存してファイルを閉じます。
5. Integration Server を再起動します。

54 保守のためのサーバの休止

- 概要 984
- 休止設定の指定 987
- Integration Server の休止 988
- 休止モードの終了 991
- 休止モード設定の展開 991

概要

Integration Server を一時的に休止すると、サーバへのアクセスが無効になるため、サーバが外部リソースに接続されていない間に保守タスクを実行できます。休止モードでは、既に進行中であったすべての要求は完了することを許可されますが、サーバへの新しい受信要求はブロックされます。JDBC プールへの接続や LDAP またはセントラルユーザディレクトリによる接続などの送信接続試行は、開いたままになります。

Integration Server が休止モードになるときに実行される処理

Integration Server が休止モードになると、以下のアセットおよびアクティビティは無効化または一時停止されます。

- **ポート**。Integration Server によって、診断ポートと休止モードを開始および終了するために使用されるポートを除くすべてのポートは無効化されます。

  サーバを休止するために使用するポートの指定の詳細については、987 ページの「休止設定の指定」を参照してください。

- **JMS メッセージング**。Integration Server によって、すべての JMS 接続エイリアスは無効化され、SOAP-JMS トリガーおよび Web サービスエンドポイントトリガーを含むすべての JMS トリガーは一時停止されます。これにより、新しいメッセージの抽出および処理は回避されます。実行中のトリガーサービスはすべて、完了するまで実行が試みられます。Integration Server が JMS トリガーを一時停止する方法の詳細については、917 ページの「コンシューマエラー後に JMS トリガーが自動的に有効化されるように Integration Server を設定する」を参照してください。

- **MQTT メッセージング**。Integration Server によって、すべての MQTT 接続エイリアスは無効化され、すべての MQTT トリガーは一時停止されます。これにより、新しいメッセージの抽出および処理は回避されます。実行中のトリガーサービスはすべて、完了するまで実行が試みられます。Integration Server が MQTT トリガーを一時停止する方法の詳細については、MQTT トリガーの状態の変更を参照してください。

- **webMethodsメッセージング**。Integration Server は、メッセージングプロバイダとして Universal Messaging を指定するすべての webMethods メッセージング接続エイリアスを無効にします。Integration Server はまた、webMethods messaging triggerのドキュメントの抽出と処理を一時停止します。これには、Universal Messaging、Broker からのドキュメント、または Integration Server からローカルにパブリッシュされたドキュメントを受信する webMethods messaging triggerが含まれます。実行中のトリガーサービスはすべて、完了するまで実行が試みられます。Integration Server が webMethods messaging triggerのドキュメント抽出を一時停止する方法の詳細については、885 ページの「ドキュメント抽出の一時停止と再開」を参照してください。Integration Server が webMethods messaging triggerのドキュメント処理を一時停止する方法の詳細については、894 ページの「ドキュメント処理の一時停止と再開」を参照してください。

- **パッケージ**。Integration Server によって、WmRoot および WmPublic を除くすべてのパッケージは無効化されます。サーバでは、パッケージとそのリソースをアンロードし、パッケージのシャットダウンサービスを完了し、メモリ内のパッケージを無効化します。

サーバをアクティブモードから休止モードに切り替えるときに、サーバがパッケージを無効化する前に待機する時間 (分) を指定できます。サービスが実行中の場合、この待機時間によって、サービスが存在するパッケージまたはそのサービスの依存サービスが存在するパッケージが無効化される前に、サービスが完了する可能性があります。待機時間を指定しない場合、Integration Server によってパッケージは直ちに無効化されます。

- **スケジュールされているユーザタスク**。Integration Server によって、スケジューラは一時停止され、既に実行中のスケジュールされているユーザタスクの完了は許可され、開始されていないスケジュールされているユーザタスクは一時停止されます。実行中のユーザタスクが、そのタスクの依存サービスが無効になる前に完了しない場合、そのタスクは未解決の依存性のためにエラーで終了します。
- **保証付きデリバー**。Integration Server によって、受信保証付きデリバージョブマネージャおよび送信保証付きデリバージョブマネージャはシャットダウンされ、ドキュメントの受信保証付きデリバーと送信保証付きデリバーは両方とも停止されます。送信デリバーの場合、watt プロパティ watt.tx.disabled が True に設定され、送信要求での保証付きデリバーの使用は無効化されます。

  注:
  サーバが休止モードになるときに受信 Job Manager と送信 Job Manager は強制的にシャットダウンされるため、処理中のトランザクションが失敗する可能性があります。サーバが休止モードになるときにトランザクションの処理をサブミットしないようにします。

  サーバが休止モードの間に受信保証付きデリバー要求を受信する場合は、最初に watt プロパティ watt.server.tx.heuristicFailRetry が True に設定されていることを確認する必要があります。次に、サービス pub.tx:init を使用して受信 Job Manager を起動します。ジョブストア内の PENDING 状態のトランザクションのサービスがサーバによって再実行されます。

  サーバが休止モードの間に送信トランザクションをサブミットする場合は、最初に watt プロパティ watt.tx.disabled を False に設定する必要があります。次に、サービス pub.tx:resetOutbound を使用して送信 Job Manager を起動します。
- **クラスタリング**。Integration Server は、 *Integration Server_directory* \instances*instance_name*\config\Caching ディレクトリ内のすべてのキャッシュマネージャをシャットダウンし、クラスタを終了します。

以下のアクティビティは、サーバが休止モードのときに続行されます。

- **アクティブなタスク**。Integration Server は、Integration Server が休止モードに入る前に開始されたユーザタスクを完了します。[**サーバ**] > [休止] ページの [**データベースと通信するシステムタスクを無効にする**] オプションが選択されている場合、データベース関連のシステムタスクも無効になります。
- **監査ログ**デフォルトでは、休止モードの間、データベースへの監査ログレコードの書き込みはサーバで続行されます。ただし、[**サーバ**] > [休止] ページの [**監査ログを無効にする**] オプションが選択されている場合、監査ログも無効になります。
- **Enterprise Gateway**内部サーバとして機能する Integration Server が休止されると、サーバは、Enterprise Gateway Server への既存の接続を切断し、サーバが休止モードになる前に開始されたユーザタスクを完了します。Enterprise Gateway Server として機能する Integration Server が休止されると、サーバの外部ポートおよび登録ポートは無効になり、内部サーバへ

の既存の接続は切断され、内部サーバから Enterprise Gateway Server に対して行われた要求はタイムアウトします。

Enterprise Gateway のシナリオでは、以下のいずれかを実行することで、2 つのサーバ間の不要な接続試行を回避できます。

- Enterprise Gateway Server と内部サーバを同時に休止します。
- 休止モードにされていないサーバの Enterprise Gateway 登録ポートを無効にします。

webMethods Enterprise Gateway の詳細については、593 ページの「webMethods Enterprise Gateway の設定 」を参照してください。

休止モードで実行できるタスク

Integration Server が休止モードの間に、以下を実行できます。

- パッケージの作成、変更、展開

  注:

  - パッケージは、休止モードで展開する前に有効にする必要があります。
  - [**サーバ**] > [**休止**] ページの [**警告と通知を無効にする**] オプションを選択し、サーバが休止モードである場合、警告は発行されず、警告および通知サービスを実行すると、例外がスローされます。

- サーバの設定の変更
- ポートの状態、JMS 接続エイリアス、webMethods メッセージング接続エイリアス、パッケージ、スケジュールされているユーザタスクの変更
- クラスタリング設定の変更 (Terracotta Server Array URL の変更など)
- キャッシュおよびキャッシュマネージャの設定の更新
- 修正のインストールおよびその他の保守タスクの実行
- サービスのデバッグ

Integration Server が休止モードを終了するときに実行される処理

Integration Server が休止モードを終了すると、以下が実行されます。

- 休止セッション中にアセットに対して行われた状態変更は保持されます。たとえば、サーバが休止モードになる前に無効化されたパッケージを有効化した場合、サーバが休止モードを終了するとパッケージは有効なままです。パッケージを作成して無効化した場合、サーバが休止モードを終了するとパッケージは無効なままです。
- 変更されていないアセットの状態は、サーバが休止モードになる前の状態に復元されます。たとえば、サーバが休止される前に、スケジュールされているユーザタスクが一時停止された場合、サーバが休止モードを終了するとタスクは一時停止のままです。
- [**サーバ**] > [**休止**] ページで無効にされた機能を有効にします。

- タスクスケジューラを再開します。
- クラスタリングアクティビティを再開します。

  休止モードを終了するときにクラスタを結合できないようなエラーが Integration Server で発生した場合、Integration Server は休止モードを再度開始しません。代わりに、Integration Server はクラスタが解除されたスタンドアロンサーバとして実行します。Integration Server は、クラスタを結合しようとしたときに発生したすべてのエラーをレポートに記録します。このレポートは、休止モードを終了した後で Integration Server Administrator に表示されます。レポートおよびエラーログは、根本的な設定エラーを修正するのに役立ちます。その後にクラスタを再結合するには、Integration Server を再起動するか、または休止モードをいったん開始してから終了します。
- サーバが休止モードの間に開始しなかった場合は、受信保証付きデリバー Job Manager と送信保証付きデリバー Job Manager が開始されます。また、watt プロパティ watt.tx.disabled が False に設定され、ドキュメントの受信保証付きデリバーと送信保証付きデリバーが再開されます。

休止設定の指定

Integration Server を休止モードで初めて起動または再起動する前に、サーバで休止モードの開始および終了に使用されるポートを指定する必要があります。このポートと診断ポートのみが、サーバが休止モードになったときに有効なままのポートです。サーバが休止モードのときに、休止ポートまたは診断ポートを無効にすることはできません。

注:
障害時回復セットアップの一部として、Integration Server で読み取り専用データベースを使用する場合、[**サーバ**] > [**休止**] ページで次のオプションを選択します。

- [監査ログを無効にする]
- [データベースと通信するシステムタスクを無効にする]
- [警告と通知を無効にする]
- [Statistics Data Collector を無効にする]

また、次のように、休止モードを開始および終了することをお勧めします。

- 休止モードを開始した後、読み取り専用操作用のデータベースを有効にする。
- 読み取り書き込み操作用のデータベースを有効にした後、休止モードを終了する。

このモードでは、Integration Server はスタンバイモードでのみ実行され、トランザクションはサポートされません。

注:
休止ポートは、WmRoot または WmPublic パッケージ内に存在する必要があります。ポートは有効である必要もあり、「allow」または「allow+」特権が必要です。休止ポートが指定されていないか、一時停止または無効化されているか、あるいは「allow」または「allow+」特権がない場合、Integration Server は休止モードになることができません。

休止ポートを指定するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. [**サーバ**] > [休止] に移動します。

3. [**休止ポートを選択**] テーブルに、[**現在の休止ポート**] が表示されます。[**新しい休止ポートを選択**] ドロップダウンリストから異なるポートを選択できます。

   注:
   このリストには、診断ポート、または webMethods Enterprise Gateway 設定で内部サーバ用に作成された HTTP/HTTPS ポートは含まれていません。

4. [**休止モードに開始するときに実行する追加タスク**] テーブルで、休止モードで無効にするアクティビティを選択します。

   - [監査ログを無効にする]
   - [データベースと通信するシステムタスクを無効にする]
   - [警告と通知を無効にする]
   - [Statistics Data Collector を無効にする]

5. [**変更内容の保存**] をクリックします。

   ページ上部のメッセージにより休止ポートおよび設定が確認されます。選択した休止設定が保存されます。

サーバを休止するには、988 ページの「Integration Server の休止 」の説明に従って、休止モードを開始します。

Integration Server の休止

Integration Server を休止するには、コマンドプロンプトから休止モードでサーバを起動するか、Integration Server Administrator でサーバをアクティブモードから休止モードに切り替えるか、またはIntegration Server Administrator から休止モードでサーバを再起動します。

また、クラスタのメンバーである Integration Server は、エラーが原因でサーバで起動時にクラスタを結合できない場合に休止モードを開始するように設定できます。そのためには、クラスタを設定するときに、[**起動エラー時のアクション**] を [**スタンドアロンで休止モードに入る**] **Integration Server**に設定します。Integration Server が起動時にクラスタに接続できないときに自動で休止モードを開始する方法の詳細については、『*webMethods Integration Server Clustering Guide*』を参照してください。

注:
Integration Server を休止モードで初めて起動する前に、サーバで休止モードの開始および終了に使用されるポートを指定する必要があります。詳細については、987 ページの「休止設定の指定」を参照してください。

コマンドプロンプトから休止モードでのサーバの起動

`-quiesce` スイッチを設定することによって、休止モードでサーバを開始できます。その他の startup コマンドスイッチについては、40 ページの「コマンドプロンプトからのサーバインスタンスの起動」を参照してください。

Integration Server Administrator からのアクティブなサーバの休止

Integration Server Administrator を使用して、アクティブな Integration Server を休止モードに切り替えることができます。

Integration Server Administrator からアクティブなサーバを休止するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. > [**休止モードを開始する**] をクリックします。

   注:
   [**休止モードを開始する**] オプションは、Integration Server がアクティブモードの場合にのみ使用できます。
3. 休止モードに切り替える前に、サーバが待機する時間 (分) を確認し指定できます。
   a. 数値を入力して、サーバが待機する時間 (分) を指定します。

      注:
      デフォルトは 0 で、これはサーバでパッケージが直ちに無効化されることを意味します。
   b. [**OK**] をクリックします。
4. 休止ポート以外のポートから Integration Server Administrator を実行している場合は、Integration Server Administrator を停止し、休止ポートを使用して起動します。

   Integration Server には、サーバが休止モードで実行されていることを示す [**休止レポート**] ページが表示され、一時停止または無効になったアセットとアクティビティのリストが表示されます。

注:
サーバが休止モードになるときにアセットまたはアクティビティを無効化または一時停止できない場合、警告メッセージが表示されて休止プロセスは続行されます。警告で示された条件が、実行しようとしている保守タスクに干渉する場合は、警告で示された問題を解決してからサーバを休止モードで再起動します。

Integration Server Administrator からの休止モードでのサーバの再起動

アクティブまたは休止した Integration Server を Integration Server Administrator から休止モードで再起動できます。

Integration Server Administrator から休止モードでサーバを再起動するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. > [**シャットダウンまたは再起動**] をクリックします。
3. [**シャットダウンと再起動**] ページで、サーバを待機後に再起動するか、直ちに再起動するかを選択します。
   - [**すべてのクライアントセッションが終了した後**]サーバを何分後に再起動するか指定します。このオプションを選択すると、サーバでユーザアクティビティの監視が開始され、すべての非管理者セッションが終了するか、指定した時間が経過するかのいずれか早い方の時間に、サーバが自動的に休止モードで再起動します。
   - [**即時**]このオプションを選択すると、サーバとすべてのアクティブなセッションは直ちに終了します。サーバは休止モードで再起動します。

注:
[**サーバ**] > [**休止**] ページで選択したアセットおよびアクティビティは、サーバを再起動しても無効です。

4. [**休止モードで再起動**] をクリックします。

5. サーバを再起動するかどうかの確認を求められたら、[**OK**] をクリックします。

6. 休止ポート以外のポートから Integration Server Administrator を実行している場合は、Integration Server Administrator を停止し、休止ポートを使用して起動します。

   Integration Server によって、Integration Server Administrator のすべてのページの上部に、サーバが休止モードで実行されていることを示すメッセージが表示されます。

   注:
   サーバが休止モードになるときにアセットまたはアクティビティを無効化または一時停止できない場合、警告メッセージが表示されて休止プロセスは続行されます。警告で示された条件が、実行しようとしている保守タスクに干渉する場合は、警告で示された問題を解決してからサーバを休止モードで再起動します。

休止モードの終了

Integration Server が休止モードで実行されている場合、Integration Server Administrator を使用して休止モードを終了し、通常の操作を再開できます。

休止モードを終了するには

1. Integration Server Administrator を開いていない場合は、それを開きます。

2. > [**休止モードを終了する**] をクリックします。

   注:
   [**休止モードを終了する**] リンクは、Integration Server が休止モードの場合にのみ使用できます。

3. サーバで休止モードを終了するかどうかの確認を求められたら、[**OK**] をクリックします。

   休止モードの終了後、Integration Server Administrator では、無効になったアセットおよびアクティビティが有効にされ、サーバが休止モードを終了した時点でのさまざまな操作の状態を示すレポートが表示されます。

休止モード設定の展開

休止ポートを指定すると、Integration Server によってポートエイリアスは休止設定ファイル (quiesce.cnf) で管理されます。このファイルは *Integration Server_directory* \instances*instance_name*\config ディレクトリにあります。quiesce.cnf ファイルは展開可能なアセットです。

webMethods Deployer を使用して quiesce.cnf ファイルを別の Integration Server に展開する場合は、以下の点に留意してください。

- 展開する休止ポートは、WmRoot または WmPublic パッケージ内に存在する必要があります。その他のすべてのパッケージはサーバが休止モードになるときに無効化されるため、休止ポートが WmRoot または WmPublic 以外のパッケージ内に存在する場合、ターゲットサーバは休止モードになることができません。
- Integration Server と共にインストールされるパッケージ (WmRoot や WmPublic など) は、展開可能なアセットではありません。quiesce.cnf ファイルを展開セットの一部として含めると、これらのパッケージは Deployer によって未解決の依存性として識別されます。この状況では、[**無視**] をクリックして依存性検査を回避します。
- ターゲットサーバで休止ポートが指定されていない場合、ターゲットサーバでは quiesce.cnf で指定されているポートが使用されます。このポートがターゲットサーバに存在することを確認するか、展開可能なアセットのリストにポートを追加します。ポートのタイプは HTTP または HTTPS である必要があります。
- Deployer でのマッピングステップ中に休止ポートエイリアスの値を置換できます。

Deployer を使用したアセットの展開の詳細については、『*webMethods Deployer User's Guide*』を参照してください。

55 Integration Server の診断

- はじめに 994
- 診断ポートの設定 994
- 診断ユーティリティの使用方法 995
- セーフモードでの Integration Server の起動 998
- サーバが自動的にセーフモードに移行した場合 999
- スレッドダンプの生成 1000

はじめに

この章には、Integration Server のトラブルシューティングを行ったり、サーバからの診断データを管理したりするサーバ管理者向けの情報が記載されています。診断データとは、設定、動作およびログに関する Integration Server からの情報です。この情報は、サーバが応答しなくなったり、回復できなくなったりした場合に役立ちます。

トラブルシューティング作業を簡単にするため、Integration Server には以下の機能が用意されています。

- **診断ポート**専用のスレッドプールを使用する特別なポートです。
- **診断ユーティリティ**Integration Server から重要な診断データを取り出す特別なサービスです。
- **セーフモードスイッチ**Integration Server の起動方法の 1 つですが、サーバは外部のリソースに接続しません。
- **スレッドダンプ**Java 仮想マシン (JVM) 内で現在実行中のスレッドおよびプロセスに関する情報を含むログを生成する機能で、Integration Server の問題の診断に役立ちます。

診断ポートの設定

診断ポートは、HTTP 経由でサブミットされた要求を処理するために専用スレッドプールのスレッドを使用する特殊なポートです。このポートは、サーバがサーバスレッドプールの代わりに診断スレッドプールを使用することを除き、通常の HTTP ポートと同じように動作します。

独立したスレッドプールを保持することによって、このポートではサーバが応答しなくなってもトラブルシューティングを行うことが可能です。たとえば、サーバが最大スレッド数に達すると、Integration Server Administrator を開くことができなくなります。このため、スレッドが使用できない理由を調べるための情報にアクセスすることができません。また、他のサーバリソースを解放することも不可能です。診断スレッドプールからのスレッドを使用すると、診断ポートを使用して Integration Server Administrator を開くことができます。

Integration Server をインストールすると、診断ポートが自動的に 9999 に作成されます。別のポートが 9999 で動作している場合、サーバは Integration Server が起動されたときに診断ポートを無効にします。診断ポートを有効にするには、ポート番号を編集する必要があります。ポート設定の編集方法については、228 ページの「ポートの編集」を参照してください。各 Integration Server には、1 つの診断ポートしか存在できません。

診断スレッドプールの設定

Integration Server Administrator では、診断スレッドプールのスレッド数を設定することができます。最大スレッド数に達するまで、サーバは必要に応じてスレッドをプールに追加します。サーバは、この最大数に達すると、実行中のプロセスが完了してスレッドがプールに戻されるまで待機してから、次のプロセスを開始します。

診断スレッドプールにスレッド優先順位を設定することもできます。診断スレッドの優先順位によって、さまざまなスレッドから JVM が要求を受け取ったときの実行順序が決まります。数が大きいほど、優先順位が高くなります。JVM は、さまざまなスレッドから要求を受け取ると、

優先順位が高いスレッドから実行します。このため、診断スレッドプールのスレッドに高い優先順位を割り当てることによって、専用のスレッドプールを利用し、Integration Server Administrator にアクセスしやすくすることができます。

診断ポートへのアクセス

Administrators グループのユーザだけが、診断ポートを通じてサーバにアクセスすることができます。http://<*hostname*>:<diagnostic port> から Integration Server Administrator にアクセスできます。この「*hostname*」は、Integration Server をホストするコンピュータで、「*diagnostic port*」は診断ポートの番号です。ユーザ名とパスワードの入力を求めるメッセージの後に、サーバは Integration Server Administrator を表示します。診断ポートには HTTP 経由でしかアクセスできないため、データとパスワードは暗号化されないクリアテキストとして渡されます。

診断ポートでは、Administrators ACL で定義されているサービスにのみアクセスすることができます。Software AG では、デフォルトのアクセス設定を変更しないことをお勧めします。

注:
Software AG では、外部のユーザが診断ポートおよび診断ユーティリティにアクセスできないようにすることを強くお勧めします。LDAP ユーザは診断ポートにアクセスしてはなりません。

診断ユーティリティの使用方法

診断ユーティリティを使用すると、設定、動作およびログに関するデータを Integration Server から収集できます。診断ユーティリティを使用して、インストールされているパッケージおよび Integration Server に当てられた修正のリストを表示することもできます。診断ユーティリティは wm.server.admin:getDiagnosticData という名前の Integration Server サービスです。WmRoot パッケージに入っています。このサービスには、Administrators グループのメンバーしかアクセスできません。このユーティリティはトラブルシューティングを行うために診断ポート経由で実行しますが、診断データを定期的に収集するために HTTP ポートまたは HTTPS ポートで使用することも可能です。このサービスには、Integration Server Administrator の [製品情報] ページの [**診断の取得**] ボタンからアクセスできます。

診断ユーティリティは一時的に diagnostics_*hostname_port_yyyyMMddHHmmss*.zip ファイルを *Integration Server_directory* \instances*instance_name*\logs ディレクトリに作成し、情報を収集したときに、この .zip ファイルに書き込みます。また、Integration Server のパッケージおよびパッケージの更新をリストする config\PackagesAndUpdates.txt ファイルも含まれています。

ファイルシステムに十分な領域がないなど、.zip ファイルの作成に問題がある場合は、テキストファイルを返します。このテキストファイルでは、設定および動作に関するデータが別個のセクションに分かれており読みやすくなっています。.zip ファイルとは異なり、このテキストファイルにはログに関するデータは含まれません。

.zip ファイルには、config ディレクトリに PackagesAndUpdates.txt という名前のファイルがあります。このファイルは、Integration Server のパッケージおよびパッケージの更新をリストします。

診断ユーティリティのパフォーマンス

Integration Server から、および場合によっては、階層化された製品から、大量のデータのログを記録する場合は、診断ユーティリティの動作が遅くなることがあります。パフォーマンスを高めるには、*maxLogSize* 値を指定するか、または `watt.server.diagnostic.logperiod` パラメータを設定して、診断ツールが返すデータの量を制限します。

wm.server.admin:getDiagnosticData サービスの *maxLogSize* パラメータでは、diagnostic_data.zip ファイルに書き込まれるログファイルのサイズ制限を設定します。ログファイルのサイズが指定された *maxLogSize* を超えた場合、診断ユーティリティは .zip ファイルにログを記録せず、diagnosticwarning.txt ファイルに記録します。diagnosticwarning.txt ファイルには、*maxLogSize* 値を超えたすべてのログファイルがリストされます。このファイルは .zip ファイルの logs ディレクトリにあります。

注:
ブラウザから診断ユーティリティを実行するときのみ、*maxLogSize* 入力パラメータを使用できます。診断ユーティリティを Integration Server Administrator から実行するときは、ログサイズを制限できません。詳細については、998 ページの「診断ユーティリティサービスの実行」を参照してください。

ログに記録する期間を指定するには、`watt.server.diagnostic.logperiod` パラメータを使用します。デフォルトでは 6 時間に設定されています。このプロパティを 0 (ゼロ) に設定すると、このユーティリティはログファイルを返さず、設定ファイルおよびランタイムデータファイルのみを返します。ユーティリティが返すログ情報は、ログをどのように保存するかによって異なります。ログをデータベースに保存する場合は、診断ユーティリティは、指定した時間数に一致する数のログエントリを返します。ログをファイルシステムに保存する場合は、指定した時間数の間だけでなく、その日のログ全体を返します。サーバ設定パラメータの設定方法については、146 ページの「拡張設定の使い方」を参照してください。

診断ユーティリティが返すログ情報は、階層化された製品からのログが.zip ファイルに含まれるかどうかによっても異なります。階層化された製品は、ActiveTransfer、Trading Networks Server など、Integration Server 上で実行される製品です。階層化された製品は、ユーティリティでその製品ログを登録することにより、診断ユーティリティに製品固有のログファイルを含めることができます。

診断アーカイブにある監査ログファイルのサイズ制限を指定するには、`watt.server.diagnostic.logFiles.maxMB` パラメータを使用します。Integration Server では、各監査ログファイルを収集するとき、要求されたログ期間のログファイルの合計サイズを計算します。ログ期間で特定の監査ログのログファイルの合計サイズが `watt.server.diagnostic.logFiles.maxMB` の値を超える場合、診断アーカイブにその監査ログファイルは含まれません。サイズ制限は、階層化された製品によって作成されたログファイルを含む、診断アーカイブに含まれるすべてのログファイルに適用されます。

以下の例について考えます。

例 1

- `watt.server.diagnostic.logFiles.maxMB` が 250 であり、`watt.server.diagnostic.logperiod` が 6 です。

- 直前の 6 時間を対象とする 2 つの WMSESSION ログファイルがあります。
- 2 つの WMSESSION ログファイルの合計サイズは 250 MB を超えています。

結果: 診断データアーカイブにセッション監査ログデータは含まれません。

例 2

- `watt.server.diagnostic.logFiles.maxMB` が 300 であり、`watt.server.diagnostic.logperiod` が 8 です。
- 直前の 8 時間を対象とする 1 つの WMSERVICE ログファイルがあります。
- WMSERVICE ログファイルのサイズは 300 MB 未満です。

結果: 診断データアーカイブにサービス監査ログデータは含まれます。

Integration Server Administrator からの診断ユーティリティの実行

Integration Server Administrator から診断ユーティリティを実行するには、以下の手順に従います。

Integration Server Administrator から診断ユーティリティを実行するには

1. Integration Server Administrator を開きます。
2. > [**製品情報**] をクリックします。
3. [**ソフトウェア**] 領域で、[**診断の取得**] をクリックします。
4. 以下のいずれかを実行できます。
   a. [**開く**] 診断データファイルを開きます。
   b. [**保存**] 診断データファイルをクライアントのファイルシステムに保存します。
   c. [**キャンセル**] この操作をキャンセルして終了します。

注:
診断データファイルを保存するか開く場合は、ファイルがクライアントのシステムで開いたり保存されたりします。Integration Server は、Integration Server を実行しているホストマシンの *Integration Server_directory* \instances*instance_name*\logs ディレクトリにコピーを自動的に保存します。

診断ユーティリティサービスの実行

Integration Server Administrator を使用せずに診断ユーティリティを実行するには、以下の手順に従います。たとえば、.zip ファイルのサイズを制限するために *maxLogSize* パラメータを使用する場合に、この方法を使用します。

Integration Server Administrator を使用せずに診断ユーティリティを実行するには

1. Web ブラウザを起動します。
2. 次の URL を入力します。

   ```
   http://<hostname>:<port>/invoke/wm.server.admin/getDiagnosticData
   ```

   この <*hostname*> はコンピュータの IP アドレスまたは名前で、<*port*> は Integration Server が動作しているポート番号です。

   注:
   .zip ファイルに含まれるログファイルのバイトサイズを制限するには、以下のように `maxLogSize` パラメータを URL に追加します。

   ```
   http://<hostname>:<port>/invoke/wm.server.admin/getDiagnosticData?maxLogSize
   =number_of_bytes
   ```

3. Administrator 特権を持つユーザ名とパスワードで Integration Server にログオンします。
4. 以下のいずれかを実行できます。
   a. [**開く**] 診断データファイルを開きます。
   b. [**保存**] 診断データファイルをファイルシステムに保存します。
   c. [**キャンセル**] この操作をキャンセルして終了します。

   注:
   診断データファイルを保存するか開く場合は、ファイルがクライアントのシステムで開いたり保存されたりします。Integration Server は、Integration Server を実行しているホストマシンの *Integration Server_directory* \instances*instance_name*\logs ディレクトリにコピーを自動的に保存します。

セーフモードでの Integration Server の起動

サーバ、またはパッケージのいずれかと外部リソースとの接続に問題があり、Integration Server が起動できない場合は、Integration Server を停止してから***セーフモード***で起動することができます。セーフモードで Integration Server を起動すると、データベースを含め、外部のリソースには接続しません。そのため、セーフモードの Integration Server は、IS コア監査機能およびプロセス監査機能に関連する監査ログデータを、外部データベースにではなく Integration Server 上のフラットファイルに書き込みます。また、セーフモードの場合、Integration Server は

WmRoot パッケージのみをロードし、他のパッケージはすべて非アクティブです。問題を診断して修正した後で Integration Server を再起動すると、Integration Server は外部データベースへの IS コア監査機能およびプロセス監査機能の監査ログ記録を再開し、すべての有効なパッケージをロードします。

重要:
セーフモードは診断またはトラブルシューティングのみを目的として使用してください。セーフモードの間は Integration Server の通常のタスクを実行したり、Designer を起動したりしないでください。Developer は予期できない結果を返します。

Integration Server が Broker またはデータベースに接続できない場合は、適切な接続パラメータを確認し必要に応じて変更します。詳細については、『*webMethods Audit Logging Guide*』を参照してください。

Trading Networks Server または webMethods SAP Adapter などのパッケージが外部のリソースに接続できない場合、Integration Server Administrator を開き、[パッケージ] > [管理] > [**パッケージをアクティブにする**] ページに移動します。[**非アクティブのパッケージ**] リストでパッケージを選択し、[**パッケージをアクティブにする**] をクリックします。Integration Server は、パッケージを Integration Server が正常に起動したときの状態にします。たとえば、パッケージが有効になっていれば、Integration Server によってパッケージがロードされ有効化されます。適切なガイドの説明を参照して、接続パラメータを確認および変更します。

セーフモードでの Integration Server の起動

セーフモードで Integration Server を起動するには

1. Integration Server に関連付けられている Java プロセス (Windows タスクマネージャなど) を停止します。

2. コマンドラインで、サーバインスタンスのホームディレクトリ (Integration Server\profiles\IS_*instance_name*) に移動し、以下のいずれかのコマンドを入力してサーバを起動します。

システム	コマンド
Windows	`bin\console.bat -safeboot` (その他のスイッチ)
UNIX	`bin/console.sh -safeboot` (その他のスイッチ)

その他のスイッチについては、40 ページの「コマンドプロンプトからのサーバインスタンスの起動」を参照してください。Integration Server Administrator を開くと、サーバがセーフモードで実行されていることを示すメッセージが表示されます。

サーバが自動的にセーフモードに移行した場合

Integration Server は、起動時にマスターパスワードまたは送信パスワードの問題を検出すると、自動的に***セーフモード***に移行します。Integration Server Administrator の [**サーバ**] > [統計情報] ページの左上隅に、次のメッセージが表示されます。

```
サーバがセーフモードで実行中です。マスターパスワードの健全性チェックが失敗しました。無効な
マスターパスワードが提供されました。
```

これらの問題の原因は、マスターパスワードファイルが 破損しているか、送信パスワードファイルが破損しているか、マスターパスワードの入力を求められたときの単純な タイプミスである可能性があります。誤ったパスワードを入力したと思われる場合は、 サーバをシャットダウンして再起動し、 正しいパスワードを入力します。以上の操作によって問題が解決されない場合は、580 ページの「起動時にマスターパスワードまたは送信パスワードに問題が生じた場合」を参照してください。

スレッドダンプの生成

Integration Server またはサブシステムが遅くなったり応答しなくなったりした場合、またはユーザが Integration Server にログインできなくなった場合は、スレッドダンプを生成して問題を診断することができます。スレッドダンプは、スレッドのブロックまたはデッドロックを引き起こす可能性のある、スレッドの競合の問題を見つけるのに役立ちます。

次のスレッドに関するスレッドダンプを生成できます。

- Integration Server が実行されている JVM
- Integration Server で実行中の個々のスレッド

これらのスレッドダンプで提供される情報を使用して、問題を解決できる場合があります。

ユーザが作成した Java サービスまたはフローサービスに関連するスレッドの問題を検出した場合は、そのスレッドをキャンセルまたは強制終了することができます。

スレッドをキャンセルすると、Integration Server はそのスレッドが保持するリソースを解放して、スレッドをスレッドプールに返します。Integration Server がスレッドをキャンセルできない場合は、スレッドを強制終了するというオプションがあります。スレッドを強制終了すると、Integration Server はスレッドを終了して、新しいスレッドをスレッドプールに追加します。サービススレッドのキャンセルおよび強制終了の詳細については、754 ページの「サービスに関連付けられたスレッドのキャンセルと強制終了」を参照してください。

次の例に、JVM スレッドダンプおよび個々のスレッドダンプを使用して、問題を診断および解決する方法を示します。

シナリオ 1: サービスが予想より長い間実行されている

1. フローサービスが非常に長い間実行されています。サービスが無限ループになっているか、または使用可能でない外部のリソースを待機していると考えられます。
2. Integration Server のプライマリポート経由でログインし、[**統計情報**] > [システムスレッド] ページに移動します。
3. [**システムスレッド**] ページで、問題のサービスに関連するスレッドを確認します。当該スレッドの個々のダンプを表示します。
4. ダンプで提供される情報を使用して、スレッドに競合の問題が発生していることを確認します。

シナリオ 1: サービスが予想より長い間実行されている

5. スレッドをキャンセルします。この操作を実行すると、サービスを完了できます。

シナリオ 2: サーバの応答がなく、ユーザがプライマリポート経由でログインできない

1. Integration Server の応答がなく、どのユーザもプライマリポート経由でログインできません。
2. 診断ポート経由でログインし、[**サーバ**] > 統計情報] > [システムスレッド] ページに移動します。
3. [**システムスレッド**] ページで、同じサービスの複数のスレッドを確認します。当該スレッドの個々のダンプを表示します。
4. ダンプで提供される情報を使用して、スレッドのキャンセルを試みます。まだ問題が発生する場合は、次に進みます。
5. スレッドの強制終了を試みます。まだ問題が発生する場合は、次に進みます。
6. JVM ダンプを実行します。
7. JVM ダンプで提供される情報を使用して、問題の原因を特定し、問題を解決できます。

次の手順に、個々のスレッドおよび JVM のダンプを生成する方法を示します。

個々のスレッドのダンプの生成

個々のスレッドに関する情報を表示するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [統計情報] に移動します。
3. [**システムスレッド**] フィールドの [**現在**] 列で、現在のスレッド数をクリックします。

   [**システムスレッド**] ページが表示されます。このページには、サーバで現在実行中の、すべてのスレッドのリストが表示されます。
4. 特定のスレッドのダンプを表示するには、そのスレッドの [**名前**] 列で、スレッド名をクリックします。

   そのスレッドのダンプが表示されます。

JVM のダンプの生成

JVM スレッドダンプを生成するには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**サーバ**] > [統計情報] に移動します。
3. [**用途**] で、[**システムスレッド**] の [**現在**] の番号をクリックします。

   [**サーバ**] > [統計情報] > [システムスレッド] ページが表示されます。

   [**システムスレッド**] テーブルに、スレッド名およびスレッドに関する情報が表示されます。
4. [**JVM スレッドダンプの生成**] をクリックします。

   [**サーバ**] > [統計情報] > [システムスレッド] > [スレッドダンプ] ページが開き、ダンプが表示されます。
5. [**システムスレッドに戻る**] をクリックして [**システムスレッド**] ページに戻ります。
6. ハードウェアまたはソフトウェアの問題によりサーバの再起動を行った場合のサーバの回復については、37 ページの「サーバの起動および停止」を参照してください。

56 エラーログによるサービス例外のデバッグ

- はじめに 1004
- 例外ログの詳細レベルの制御 1004
- エラーログの表示 1004
- エラーログの解釈 1004

はじめに

Integration Server のエラーログは、サービスによってスローされた例外に関する情報を保持します。この付録では、ログを使用して、サービス例外のデバッグに役立てる方法について説明します。

例外ログの詳細レベルの制御

`watt.server.deprecatedExceptionLogging` パラメータを設定すると、Integration Server がサービス例外をログに記録する方法および詳細度のレベルを制御できます。この付録では、このパラメータが、より詳細な例外ログを得られるデフォルト設定 (`false`) に設定されていると想定します。このパラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。

エラーログの表示

Integration Server Administrator で、[**ログ**] > [エラー] ページに移動してエラーログを表示します。エラーログのフィールドは次のとおりです。

列	[詳細]
[タイムスタンプ]	エントリがログに書き込まれた日時。
[サービス名]	例外が発生したサービスの名前。
[サービススタック]	例外が発生したサービスの親サービス。
[エラーメッセージ]	発生した例外の内容を示すメッセージ。
[スタックトレース]	例外が発生するまでの呼び出しシーケンスを示すトレース。スタックトレースデータの表示を展開するには、[**ログの表示制御**] 領域の [**スタックトレースデータの展開**] チェックボックスをオンにして、[**リフレッシュ**] をクリックします。
[ルートのコンテキスト] **[親のコンテキスト]** **[現在のコンテキスト]**	webMethods Monitor が、さまざまなログの関連エントリに接続するときに使用するコンテキスト情報。

エラーログの解釈

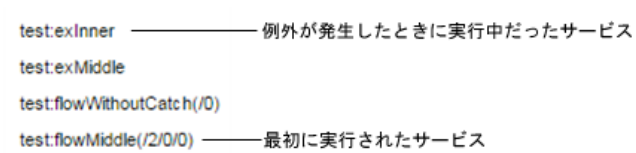
次の図に、例外が発生したステップの識別に役立つ詳細が含まれるエラーログの一部を示します。次の 2 つのアクションを実行すると、この詳細情報が Integration Server Administrator に表示されます。

- watt.server.deprecatedExceptionLogging パラメータの値をデフォルト値 (`false`) に設定する
- [**ログ**] > [エラー] に移動し、[**スタックトレースデータの展開**] を選択します。

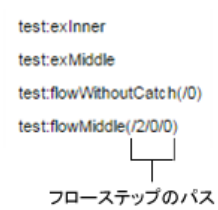
詳細な例外ログを表示した Integration Server エラーログ

サービススタックについて

[**サービススタック**] 列には、例外の発生時点までに呼び出された Java サービスおよびフローサービスが表示されます。スタックには、サービスが次の順序で表示されます。



フローサービスの場合、[**サービススタック**] 列には、例外発生時点におけるサービス内のステップを示すインデックス済みパスも表示されます。このパスの形式は `/n/n/n...` です。この `n` はフローステップの連続したインデックス、`/` はネストのレベルを表します。



このパスによって、例外が発生したステップを特定でき、フローサービスが複数回呼び出された場合またはサービスやステップが深くネストされている場合に特に有効です。たとえば、serviceA が serviceB を 3 回呼び出し、serviceB の 2 回目の呼び出しで例外が発生したとします。この場合、エラーログの [**サービススタック**] 列には、パスが `serviceA(/1/2)` と表示されます。

Step	インデックス
INVOKE serviceB	/0
SEQUENCE	/1
INVOKE serviceD	/1/0
INVOKE serviceE	/1/1
INVOKE serviceB	/1/2 （**ここで例外がスローされる**）
SEQUENCE	/2
INVOKE serviceC	/2/0
INVOKE serviceB	/2/1

エラーメッセージについて

詳細な例外ログを指定すると、エラーログの [**エラーメッセージ**] 列には、その例外に対応するエラーメッセージおよびネストした各例外のエラーメッセージを連結したメッセージが表示されます。これらのメッセージの形式は次のとおりです。

```
Outer exception message
Caused by: middle.exception.className: Middle exception message
Caused by: inner.exception.className: Inner exception message
```

Integration Server が提供する標準のエラーメッセージテキスト (「[ISS.0062.9021] "object" が NULL です。」など) に加えて、例外メッセージには、 サービス開発者がサービスのエラー処理部分で指定する、 カスタマイズしたテキストを追加することができます。

スタックトレースについて

例外の中には、他の例外がネストして含まれていることがよくあります。例外発生までの呼び出しシーケンスが [**スタックトレース**] 列に表示されるとき、その範囲は `watt.server.deprecatedExceptionLogging` パラメータに選択した設定によって次のように異なります。

- このパラメータを「`false`」 (詳細な例外ログ) に設定すると、エラーログの [**スタックトレース**] 列には、最も深いスタックトレース (問題のソースを示すスタックトレース) が表示されます。
- このパラメータを「`true`」 (基本の例外ログ) に設定すると、スタックトレースが切り捨てられることが多く、例外の原因を追跡することが難しくなります。このため、Software AG では、例外をキャッチしても再スローしないサービスを実行中でない限り、このパラメータは「`true`」に設定しないことをお勧めします。

57 Docker での Integration Server の使用

- Docker と Integration Server の概要 1008
- is_container スクリプトについて 1009
- Docker イメージのビルドの前提条件 1010
- Microservices Runtime の Docker イメージから Web サービスサポートを除外する .. 1011
- webMethods Cloud でコンシューマに公開するサービスの指定 1012
- Integration Server インスタンス用の Docker イメージのビルド 1012
- Docker イメージをオンプレミス Docker レジストリにロードする 1018
- Docker イメージをオンプレミス Docker レジストリにプッシュする 1019
- オンプレミス Docker コンテナでの Docker イメージの実行 1019
- Docker イメージを Integration Cloud にプッシュする 1025
- Integration Cloud での Docker イメージの実行 1026
- Integration Server の Docker コンテナの停止 1027
- Docker イメージでの設定変数テンプレートの使用 1027

Docker と Integration Server の概要

Docker はオープンソーステクノロジです。ユーザは Docker を使用して、アプリケーションをソフトウェアコンテナに展開できます。Docker コンテナは Docker イメージのインスタンスです。Dockerイメージはファイルシステムとランタイムパラメータを含むアプリケーションです。Docker イメージは、インストールされ設定された Integration Server インスタンスから作成した後、Docker コンテナ内で実行できます。Docker コンテナでの Integration Server の実行を簡単にするため、Integration Server ではスクリプトを使用して Docker イメージをビルドし、ビルドされた Docker イメージを、オンプレミスまたは webMethods Integration Cloud にホストされた Docker レジストリにロードまたはプッシュすることができます。

注:
Integration Server または Microservices Runtime のインスタンスの Docker イメージをビルドできます。特に指定のない限り、Integration Server と Docker の使用手順は Microservices Runtime に適用されます。

Docker 1.12.11 以降での Integration Server のサポートは、Docker がネイティブサポートを提供する Linux システム、UNIX システム、および Windows Server 2016 で利用できます。

注:
Integration Server のマニュアルでは、Docker テクノロジの知識を持っていることを前提としています。Docker およびコンテナテクノロジの詳細な説明はこのマニュアルの範囲外ですが、別途入手できます。

Docker での Integration Server の使用の推奨事項

Docker コンテナでの Integration Server の実行を選択した場合、以下のことをお勧めします。

- インストール済みで設定が完了したオンプレミス Integration Server 用の Docker イメージを作成します。イメージの作成前にサーバの設定ができるだけ完了していることを確認してください。Docker イメージの作成後、一部の設定は変更または追加できますが、できるだけ多くの設定を基本イメージで行うことをお勧めします。
- Microservices Runtime および Integration Server には、Docker イメージを作り直す必要なく、Docker コンテナ内で実行されているサーバの設定またはコンテンツの追加または変更に使用できる機能が用意されています。具体的には以下のとおりです。
  - 設定変数テンプレートは、Docker コンテナ内で実行している Integration Server または Microservices Runtime のサーバ設定を変更するのに使用できます。設定変数テンプレートは、Docker コンテナ内の Integration Server または Microservices Runtime にアセットを追加したり、以前設定されていなかったアセットプロパティを設定したりするのに使用することもできます。

    Software AG では、Docker の基本イメージに設定を追加することをお勧めします。設定変数を使用して、環境内のエンドポイントまたは設定値を変更します。

    設定変数の詳細については、『』を参照してください。
  - EXTERNALIZE_PACKAGES ENV 変数は、Docker コンテナ内で実行している Integration Server により多くのパッケージを使用可能にするために使用されます。Docker コンテナ

を開始すると、XTERNALIZE_PACKAGES ENV 変数を指定できます。この環境変数の詳細については、1283 ページの「環境変数」を参照してください。

- パッケージの自動展開により、Docker イメージを再構築する必要なく、Docker コンテナ内で実行しているサーバのカスタムパッケージをインストールまたは更新することができます。パッケージの自動展開の詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

上記の機能を使用しない場合、Integration Server の Docker イメージは変更しないように注意してください。つまり、Docker コンテナで実行されている Integration Server の設定またはコンテンツは変更しないでください。代わりに、オンプレミス Integration Server に変更を行い、Docker イメージを再作成し、その Docker イメージを Docker リポジトリにロードまたはプッシュした後、イメージに対して Docker コンテナを起動します。

注:
設定変数テンプレートおよびパッケージの自動展開機能は、追加ライセンスにより Microservices Runtime または Integration Server で使用可能です。

- Integration Cloud の Integration Server 用に作成された Docker イメージを実行する場合、Software AG では Docker イメージの作成対象の Integration Server で CSRF ガードを有効にし、設定することを強くお勧めします。CSRF ガード機能を有効にすると、Integration Server Administrator URL によるクロスサイトリクエストフォージェリ (CSRF) 攻撃が防止されます。Integration Server インスタンスで CSRF ガードを有効にし設定した後、インスタンスに対する Docker イメージを作成します。CSRF ガードの有効化と設定の詳細については、585 ページの「CSRF ガードによる Integration Server のセキュリティ確保」を参照してください。

is_container スクリプトについて

is_container.sh/bat スクリプトによって Docker とのインタラクションが簡単になります。このスクリプトは、Dockerfile の作成、Integration Server 用 Docker イメージのビルド、オンプレミスにホストされた Docker レジストリへの Docker イメージのロードまたはプッシュ、および webMethods Cloud にホストされた Docker レジストリへの Docker イメージのプッシュを行うコマンドを備えています。

is_container.sh/bat スクリプトは以下の項目の提供を目的としています。

- 顧客が webMethods Integration Cloud Container Edition を利用する方法。スクリプトは webMethods Integration Cloud でシームレスに動作します。
- Docker を自社インフラストラクチャで使用するオンプレミス Integration Server のユーザ用のスタータースクリプト。

注:
is_container.sh/bat スクリプトはスタータースクリプトとして使用することを目的としています。このスクリプトは Docker 機能をすべてサポートしているわけではありません。is_container.sh/bat スクリプトは、ユーザが独自のスクリプトを作成する際のサンプルとして参照するために用意されています。

is_container.sh/bat スクリプトは *Software AG_directory*\ *Integration Server_directory* \docker にあります。

is_container スクリプトコマンド

is_container.sh/bat スクリプトで使用できるコマンドの説明を以下の表に示します。

コマンド	説明
`createDockerfile`	Default パッケージおよび Wm パッケージのみを含む、基本 Integration Server インスタンス用の Dockerfile を作成します。
`createLeanDockerfile`	基本 Integration Server インスタンス用の Dockerfile を作成します。WmRoot、WmPublic、WmCloud などのコア Integration Server に必要な Default パッケージおよび Wm パッケージを含みます。
`createPackageDockerfile`	カスタムパッケージ用の Dockerfile を作成します。
`build`	指定された Dockerfile を使用して Docker ビルドを実行し、Integration Server の基本イメージをビルドします。
`buildPackage`	指定された Dockerfile を使用して Docker ビルドを実行し、カスタムパッケージを含むIntegration Server のイメージをビルドします。
`saveImage`	ローカル Docker レジストリのイメージを、file.path パラメータで指定された tar ファイルに保存します。
`loadImage`	file.path パラメータに指定されたイメージをローカル Docker レジストリにロードします。
`pushImage`	オンプレミス Integration Server 用に作成された Integration Server イメージを webMethods Cloud または Docker レジストリにプッシュします。
`help`	各コマンドの使用情報を表示します。

Docker イメージのビルドの前提条件

Integration Server 用の Docker イメージをビルドする前に、以下の作業を完了します。

- Integration Server をインストールし、Docker をデーモンとして起動するマシンに Docker をインストールします。
- *Installing Software AG Products* の指示に従って Linux、UNIX システム、または Windows Server 2016 に Integration Server、パッケージおよび修正をインストールし、Integration Server およびホストされた製品を設定します。
- Integration Cloud の Docker コンテナ内で Integration Server を実行する予定であり、一部のサービスをコンシューマに公開する場合、Docker イメージのビルド前にその Integration Server サービスを指定する必要があります。詳細については、1012 ページの「webMethods Cloud でコンシューマに公開するサービスの指定」を参照してください。

- Microservices Runtime の Docker イメージをビルドする場合、Web サービス機能をイメージに含めるかどうかを決定できます。

  詳細については、1011 ページの「Microservices Runtime の Docker イメージから Web サービスサポートを除外する」を参照してください。

Microservices Runtime の Docker イメージから Web サービスサポートを除外する

is_container スクリプトには、Microservices Runtime 用に作成される Docker イメージから Web サービス機能を除外するオプションがあります。Web サービスを除外すると、Docker イメージのサイズは大幅に減少します。ただし、Web サービス機能に依存する他の Microservices Runtime 機能は使用できず、実行されません。具体的には以下のとおりです。

- Web サービス機能に依存する組み込みサービスは実行できません。次のいずれかのサービスを実行しようとすると、ServiceException になります。
  - pub.soap* サービス
  - pub.utils.ws.setCompatibilityModeFalse
  - pub.client:soapClient
  - pub.client:soapHTTP
  - pub.client:soapRPC
  - *jmsTriggerType*=SOAPJMS の場合、pub.trigger:createJMSTrigger
- Docker コンテナで実行されている Microservices Runtime で、SOAP-JMS トリガーは作成できません。さらに、既存の標準 JMS トリガーは SOAP-JMS に変更できません。
- WSDL URL へのアクセスまたは Web サービスの呼び出しを要求すると、次のメッセージと共に HTTP 状態コード 404 が返されます: WebServices Support (WebServices) はインストールされた製品にありません。
- Web サービス機能に関連するサーバ設定パラメータに効力はありません。
- Web サービスエンドポイントエイリアスおよびポリシーなど、Web サービス設定は無視されます。
- Web サービス記述子を含むカスタムパッケージが、Docker コンテナで実行されている Microservices Runtime によってロードされる場合、Microservices Runtime はパッケージを部分的にロードします。Web サービス機能に依存するパッケージのアセットはロードされず、使用できません。Web サービス記述子の操作またはサービスを実行しようとすると、server.log に次のメッセージが記録されます: Web サービス記述子 *webServiceDescriptorName* をロードできません。WebServices Support (WebServices) はインストールされた製品にありません。
- Web サービス記述子または Web サービス機能に依存する他のアセットを含むシステムパッケージが Docker コンテナで実行されている Microservices Runtime によってロードされた場合、Microservices Runtime はパッケージ全体をロードします。Web サービス記述子の操

作またはサービスを実行しようとすると、server.log に次のメッセージが記録されます: Web サービス記述子 *webServiceDescriptorName* をロードできません。WebServices Support (WebServices) はインストールされた製品にありません。

webMethods Cloud でコンシューマに公開するサービスの指定

Integration Cloud の Docker コンテナ内で実行されている Integration Server サービスをコンシューマが使用できるようにするには、Integration Server 用の Docker イメージを作成する前に、追加設定を完了する必要があります。具体的には、Docker コンテナが Integration Cloud で公開する Integration Server サービスを指定する必要があります。Integration Cloud の外側からのサービス呼び出し要求は直接 Docker コンテナのサービスを呼び出すことはできません。代わりに、要求は Integration Cloud エンドポイントを経由する必要があり、そこから要求は、要求されたサービスを公開する Docker コンテナにリダイレクトされます。

Integration Cloud に公開するサービスを指定するには

1. Integration Server Administrator に移動し、**[webMethods Cloud] > [Docker サービス]** に移動します。

   [パッケージ] 列には Integration Server のすべてのパッケージの一覧が表示されます。

2. [**Dockerサービス**] 列で、カスタムパッケージからコンシューマに公開するサービスを指定します。

   以下の 1 つ以上を使用して、サービスを指定できます。

   - 以下の形式のディレクティブおよびサービスの完全修飾名

     /directive/folder.subfolder:serviceName

     例: /invoke/is.assets:getChecksums

   - サービスを呼び出す URL

     URL として、Integration Cloud が Docker コンテナのサービスに正確にマップできる一意のスニペットを指定します。

   - [**設定**] > [URL エイリアス] > [URL エイリアスの作成] を使用して作成された URL エイリアス

   カンマを使用してサービスを区切ります。

3. [**変更内容の保存**] をクリックします。

Integration Server インスタンス用の Docker イメージのビルド

Integration Server 用の Docker イメージのビルドは以下のように構成されます。

- Integration Server インスタンス用の Dockerfile を作成する。この Dockerfile には、システム (Wm*) パッケージとコア機能が含まれます。

- コア機能とシステムパッケージが入った Dockerfile を使用して、Integration Server の基本 Docker イメージをビルドする。
- Integration Server イメージに含めるカスタムパッケージに対して、上記の手順で作成した基本イメージが入った Dockerfile を作成する。
- カスタムパッケージ Dockerfile を使用して、Integration Server イメージの Docker イメージをビルドする。その結果、この Docker イメージには、コア機能、システムパッケージ、およびカスタムパッケージが含まれます。

  注:
  パッケージの自動展開を使用する場合、カスタムパッケージにイメージを作成する必要はありません。代わりに、基本イメージおよび Default パッケージを含むイメージを作成します。その後、パッケージの自動展開機能を使用して Docker コンテナで実行している Microservices Runtime または Integration Server にパッケージを追加します。パッケージの自動展開の詳細については、*Developing Microservices with webMethods Microservices Runtime.* を参照してください。

Docker コンテナで使用する Integration Server イメージをビルドする前に、1010 ページの「Docker イメージのビルドの前提条件」にリストされている前提条件を完了してください。

のローカルサービス開発機能用のローカル開発サーバとして Docker コンテナで実行される Integration Server の使用方法 (前提条件を含む) の詳細については、『*webMethods Service Development Help*』を参照してください。

注:
Integration Server によって作成された Dockerfile には、Dockerfile のイメージに使用される Integration Server インスタンスに関するメタデータが含まれます。メタデータは Dockerfile のラベルとして表示され、名前、製品バージョン、製品ビルド番号、インスタンスのプライマリポートが含まれます。

Integration Server 用の Docker イメージを作成するには

1. *Software AG_directory*\IntegrationServer\docker ディレクトリに移動します。
2. 以下のコマンドの 1 つを使用して、Integration Server インスタンスのコア用の Dockerfile を作成します。Dockerfile は *Software AG_directory* に作成されます。
   - 以下の `createDockerFile` コマンドを実行することで、JDK か JRE の選択、必要なだけの Wm パッケージ (つまり、システムパッケージや Trading Networks Server などの製品パッケージ) を指定して Dockerfile を作成します。ファイルは Software AG ディレクトリに保存されます。

     ```
     is_container.sh/bat createDockerfile [optional arguments]
     ```

引数	説明
`-Dimage.name=` *baseImageName*	オプション。ubuntu、centos:8 などの基本イメージ。 デフォルト: centos:8

引数	説明
	Windows 上で実行されている Integration Server の場合、`-Dimage.name` パラメータは次のように設定します。 `-Dimage.name=mcr.microsoft.com/windows/servercore:`*`YourOsImageTag`* ここで、*`YourOsImageTag`* は、Integration Server が実行されている OS バージョン用に作成された Windows イメージのタグです。
`-Dinstance.name=` *instance_ name*	オプション。イメージに含める Integration Server インスタンス。 デフォルト: default (default という名前のインスタンス)
`-Dport.list=` *ports*	オプション。イメージで公開するインスタンスのポートのカンマ区切りリスト。 デフォルト: Integration Server はプライマリポート、診断ポート、およびデフォルトセキュアポートに割り当てられたポート番号を使用します。
`-Dpackage.list=` *Wm packages*	オプション。イメージに含めるインスタンスの Wm パッケージのカンマ区切りリスト。 デフォルト: all (すべての Wm パッケージと Default パッケージが含まれます)
`-Dinclude.jdk=`{true|false}	オプション。イメージに Integration Server JDK (true) を含めるか JRE (false) を含めるか。 デフォルト: true **注:** Java 11 と実行されている Integration Server の場合、-Dinclude.jdk 値と関係なく、完全な JDK が使用されます。
`-Dfile.name=` *Dockerfile_name*	オプション。生成された Dockerfile のファイル名。 デフォルト: Dockerfile_IS
`-Dtarget.configuration=` *configuration_name*	オプション。Dockerfile の作成対象の Integration Server のターゲット設定。 デフォルト: このパラメータのデフォルト値はありません。 Dockerfile がローカル開発サーバとして使用される Integration Server 用である場合、`localdev` を指定します。 のローカルサービス開発機能用のローカル開発サーバとして Docker コンテナで実行される Integration Server の使用方法

引数	説明
	の詳細については、『*webMethods Service Development Help*』を参照してください。 Dockerfile が OpenShift と共に使用される Integration Server 用である場合、`OpenShift` を指定します。 `-Dtarget.configration=OpenShift` の場合、is_container.bat/sh スクリプトによって読み取り、書き込み、および実行の権限を持つルートグループが作成されます。これが必要となるのは、OpenShift が任意に割り当てられたユーザ ID を使用して Docker コンテナを実行するためです。任意のユーザによるコンテナの実行をサポートするため、ルートグループは Docker イメージのプロセスによって書き込まれる可能性があるディレクトリとファイルを所有する必要があり、ルートグループにはディレクトリとファイルへの読み取り/書き込みアクセス権が必要であり、ルートグループには実行可能なファイルの実行権限が必要です。 Dockerfile の対象である Integration Server インスタンスが Integration Cloud に移行される場合、`wmcloud` に設定します。 このパラメーターを「`wmcloud`」に設定すると、is_container スクリプトは、Integration Server で提供されているデフォルトのトラストストアを、Integration Cloud で必要とされるトラストストアで上書きします。
`-Dimage.createUser={true|false}`	オプション。Docker イメージのファイルの所有者に対して root 以外のユーザを設定します。`-Dimage.createUser` が「true」に設定されている場合、Integration Server は Docker イメージ内で sagadmin ユーザがファイルの所有権を持つ Docker イメージを作成します。このパラメータのデフォルト値は「false」です。 デフォルト: false **注:** このオプションはWindows を除くオペレーティングシステムに適用され、Docker はバージョン 17.09 以上である必要があります。 **注:** `-Dtarget.configuration=OpenShift` の場合、スクリプトによって `-Dimage.createUser` は自動的に「false」に設定されます。
`-Dexclude.components = "componentName"`	オプション。イメージから除外するコンポーネントのカンマ区切りリスト。現在除外できる唯一のコンポーネントは

引数	説明
	`WebServices` です。この引数は、Microservices Runtime 用の作成された Docker イメージにのみ適用されます。

- 以下の `createLeanDockerfile` コマンドを実行することで、JDK を含み、Integration Server が動作するのに必要な最小限のシステムパッケージ (WmRoot、WmPublic、および WmCloud) のみ含む Dockerfile を作成します。

```
is_container.sh/bat createLeanDockerfile [optional arguments]
```

オプションの引数は *-Dimage.name*、`-Dinstance.name`、`-Dport.list`、`-Dfile.name`、`-Dtarget.configuration`、`-Dimage.createUser`、および `-Dexclude.components` です。

3. 以下の `build` コマンドを実行することで、コア Dockerfile を持つ Docker イメージをビルドします。イメージはローカル Docker レジストリに保存されます。

```
is_container.sh/bat build [optional arguments]
```

引数	説明
`-Dfile.name=` *Dockerfile_name*	オプション。Docker イメージのビルドに使用する Dockerfile のファイル名。 デフォルト: Dockerfile_IS
`-Dimage.name=` *Docker_ image_name*	オプション。生成される Docker イメージの名前。 デフォルト: is:micro

4. 以下の `createPackageDockerfile` コマンドを実行することで、Docker イメージに含めるカスタムパッケージを指定して Dockerfile を作成します。カスタムパッケージの Dockerfile は *Software AG_directory*/ *Integration Server_directory* /instances/*instance name*/packages ディレクトリに保存されます。

```
is_container.sh/bat createPackageDockerfile [optional arguments]
```

引数	説明
`-Dinstance.name=` *instance_name*	オプション。ユーザ定義パッケージを含む Integration Server インスタンス。 デフォルト: default (default という名前のインスタンス)
`-Dimage.name=` *Docker_image_name*	オプション。このイメージをビルドするコア Integration Server イメージの名前。 デフォルト: is:micro

引数	説明
`-Dpackage.list=` *packages*	オプション。イメージに含めるカスタムパッケージのカンマ区切りリスト。 デフォルト: all 「all」は、パッケージのイメージに Integration Server インスタンスのすべてのカスタムパッケージを含めることを示します。Default パッケージおよび「Wm」で始まるパッケージはイメージに含まれません。 **注:** Integration Server が Integration Cloud の Docker コンテナで実行されるときに、コンシューマに公開するサービスの入っているカスタムパッケージを必ず含めてください。公開されるサービスを含むパッケージを含めなかった場合、Integration Cloud がサービスに対する要求を Docker コンテナにリダイレクトすると、サービスが見つからないというエラーが発生し、要求は失敗します。
`-Dimage.createUser={true|false}`	オプション。Docker イメージのファイルの所有者に対して root 以外のユーザを設定します。`-Dimage.createUser` が「true」に設定されている場合、Integration Server は Docker イメージ内で sagadmin ユーザがファイルの所有権を持つ Docker イメージを作成します。このパラメータのデフォルト値は「false」です。 **注:** このオプションはWindows を除くオペレーティングシステムに適用され、Docker はバージョン 17.09 以上である必要があります。
`-Dfile.name=` *Dockerfile_name*	オプション。カスタムパッケージを含む Dockerfile のファイル名。 デフォルト: Dockerfile_IS_Pkg

注:
パッケージの自動展開を使用している場合、上記のステップはスキップできます。代わりに、Docker コンテナ内で Microservices Runtime または Integration Server が実行されている場合、自動展開フォルダにマッピングされたホストマシンのフォルダに、カスタムパッケージを配置できます。詳細については、『*Developing Microservices with webMethods Microservices Runtime.*』を参照してください。.

5. 以下の `buildPackage` コマンドを実行することで、カスタムパッケージの Dockerfile を持つ Docker イメージをビルドします。イメージはローカル Docker レジストリに保存されます。

ビルドされた Docker イメージには、カスタムパッケージ、システムパッケージ、および Integration Server のコア機能が含まれます。

```
is_container.sh/bat buildPackage [optional arguments]
```

引数	説明
`-Dfile.name=` *Dockerfile_name*	オプション。カスタムパッケージを含む Dockerfile のファイル名。 デフォルト: Dockerfile_IS_Pkg Docker ビルドでは、指定されたインスタンスのパッケージディレクトリ *Software AG_directory*/*Integration Server_directory* /instances/*instance name*/packages/ にある指定された名前のファイルを使用します。
`-Dimage.name=` *Docker_image_name*	オプション。カスタムパッケージを含む生成される Docker イメージの名前。 デフォルト: is:microPkg

6. オプションとして、必要に応じて、以下の 1 つ以上のコマンドライン引数を実行します。

   - `saveImage`: Integration Server イメージをファイルに保存します。`saveImage` の詳細については、1018 ページの「Docker イメージをオンプレミス Docker レジストリにロードする」を参照してください。
   - `loadImage`: Docker レジストリにイメージをロードします。`loadImage` の詳細については、以下を参照してください。1019 ページの「Docker イメージをオンプレミス Docker レジストリにプッシュする」
   - `pushImage`: オンプレミス Integration Server 用に作成された Integration Server イメージを Integration Cloud または Docker レジストリにプッシュします。Docker レジストリまたは Integration Cloud へのイメージのプッシュの詳細については、1019 ページの「Docker イメージをオンプレミス Docker レジストリにプッシュする」および以下を参照してください。1025 ページの「Docker イメージを Integration Cloud にプッシュする」

Docker イメージをオンプレミス Docker レジストリにロードする

Integration Server 用の Docker イメージをオンプレミス Docker レジストリにロードするには、最初にイメージを tar ファイルとして保存し、次に tar ファイルをレジストリにロードします。

オンプレミス Docker レジストリへの Docker イメージのプッシュの詳細については、1019 ページの「Docker イメージをオンプレミス Docker レジストリにプッシュする」を参照してください。

Docker イメージを保存し、オンプレミス Docker レジストリにロードするには

1. 以下の saveImage コマンドを実行することで、Docker イメージを tar ファイルとして保存します。デフォルトで、tar ファイルは *Software AG_directory*/ *Integration Server_directory* /docker/images ディレクトリに保存されます。

```
is_container.sh/bat saveImage -Dimage.name=input Docker image
[-Dfile.path=full path to output
tar file]
```

2. 以下の loadImage コマンドを実行することで、tar ファイルの Docker イメージを Docker レジストリにロードします。

```
is_container.sh/bat loadImage [-Dfile.path=full path to input tar
file]
```

Docker イメージをオンプレミス Docker レジストリにプッシュする

pushImage コマンドを使用して、オンプレミス Integration Server 用に作成された Dockerイメージをオンプレミス Docker レジストリにプッシュできます。

Docker イメージをオンプレミス Docker レジストリにプッシュするには

- 以下の pushImage コマンドを実行します。

```
is_container.sh/bat pushImage required arguments[optional argument]
```

引数	説明
-Duser= *userID*	レジストリにアクセスするユーザ ID。
-Dpassword= *password*	オプション。レジストリにアクセスするパスワード。この引数を使用してパスワードを指定しなかった場合、pushImage コマンドの実行時にパスワードの入力が求められます。
-Dserver= *registry_URL*	レジストリの URL。 例: docker.io
-Drepository.name= *repository_name*	イメージのプッシュ先のリポジトリ名。
-Dimage.name= *Docker_ image_name*	Integration Server イメージ is:microPkg など、プッシュする Docker イメージの名前。

オンプレミス Docker コンテナでの Docker イメージの実行

docker run コマンドを使用して、Integration Server 用に作成した Docker イメージなどの Docker イメージをオンプレミス Docker コンテナで実行します。Docker イメージが Integration Server 用である場合、Docker イメージを実行すると、Integration Server が起動します。

docker run コマンドを使用する場合、Docker コンテナをホストするマシンのポートを明示的に Docker コンテナの公開ポートにマップできます。または、Docker は公開ポートを Docker ホストマシンの任意の使用可能なポートにマップできます。Integration Server 用の Docker コンテナの場合、公開ポートにはプライマリポートと診断ポートが含まれます。Integration Server Administrator を使用して、Docker コンテナで実行されている Integration Server に接続する場合、ホストマシンポートをコンテナの Integration Server プライマリポートに明示的に割り当てる必要があります。ホストマシンポートと Integration Server ポートとの関連付けを Docker に行わせると、Docker が Integration Server プライマリポートをどのホストマシンポートと関連付けたかわかりません。同様に、診断ポート経由で Integration Server Administrator にアクセスできるようにする場合、コンテナで公開されている診断ポートにホストマシンのポートを明示的にマップする必要があります。

注:
多くのコンテナ展開ソリューションは、コンテナのコンソールログ STDOUT および STDERROR を表示する機能を備えています。Docker コンテナで Integration Server の Docker イメージを実行すると、Integration Server はサーバログを server.log ファイルだけでなくコンソールにも書き込みます。

オンプレミス Docker コンテナで Docker イメージを実行するには

- 次のコマンドを実行します。

```
docker run -d --name Docker_container_name
              -p [host_primary_port:]primary_port
              -p [host_diagnostic_port:]diagnostic_ port              -p
[host:]other_exposed_ port environment_variables
                  Docker_image
```

プレースホルダ	指定内容
Docker_container_name	Docker イメージを実行する Docker コンテナの名前。たとえば、IS_Default です。
host_primary_port	オプション。Docker コンテナの公開ポート（Integration Server のプライマリポート）に明示的にマップするコンテナホストマシンのポート。*host_primary_port* を指定しない場合、Docker コンテナはコンテナホストマシンの任意の使用可能なポートを *primary_port* にマップします。
primary_port	Integration Server 上のプライマリポートのポート番号。
host_diagnostic_port	オプション。Docker コンテナの公開ポート（Integration Server の診断ポート）に明示的にマップするコンテナホストマシンのポート。*host_diagnostic_port* を指定しない場合、Docker コンテナはコンテナホストマシンの任意の使用可能なポートを *diagnostic_port* にマップします。
diagnostic_port	Integration Server 上の診断ポートのポート番号。

プレースホルダ	指定内容
host_port	オプション。Docker コンテナの公開ポートに明示的にマップするコンテナホストマシンのポート。*host_port* を指定しない場合、Docker コンテナはコンテナホストマシンの任意の使用可能なポートを *other_exposed_port* にマップします。
other_exposed_port	Integration Server 上の公開ポートのポート番号。
environment_variables	使用する環境変数の名前=値ペア。Docker に使用する Integration Server 環境変数の一覧については、1283 ページの「環境変数」を参照してください。
Docker_image	オンプレミス Docker コンテナで実行する Docker イメージの名前。たとえば、is:microPkg です。

例

Integration Server 用の Docker イメージでポート 5555 と 9999 を公開する場合の例です。

この例で、コマンドにはコンテナホストマシンのポートが含まれません。したがって、Docker コンテナは公開ポートをコンテナホストマシンの使用可能なポートに自動的にマップします。

```
docker run -d --name IS_Default -p 5555 -p 9999 is:microPkg
```

以下の例で、コマンドはコンテナホストマシンで使用できる ポートを Docker コンテナの公開ポートに明示的に マップします。

```
docker run -d --name IS_Default -p 34678:5555 -p 34679:9999 is:microPkg
```

注:
コマンド `docker ps -a` を実行することで、Docker コンテナの実行中のインスタンスを確認できます。

コンソールへのサーバログの書き込み

多くのコンテナ展開ソリューションは、コンテナのコンソールログ STDOUT および STDERROR を表示する機能を備えています。Docker コンテナで Integration Server の Docker イメージを実行すると、Integration Server はサーバログを server.log ファイルだけでなくコンソールにも書き込みます。STDOUT に書き込まれているサーバログメッセージは「ISSERVER」識別子を含み、サーバログメッセージをコンソールに書き込まれた他のメッセージから区別するのに役立ちます。

Docker には、実行中のコンテナからの情報取得に役立つログドライバが含まれています。Docker ソリューションに json-file ログドライバまたは journald ログドライバを使用すると、ドライバは STDOUT に書き込まれる情報を取得し、ファイルに書き込みます。`docker logs` コマンドを実行すると、そのファイルの内容が表示されます。ログに使用されるファイルのサイズは増加する可能性があります。ファイルのサイズを制限し、ディスクのオーバーフローの問題を避けるため、ログドライバオプションの使用を検討することをお勧めします。

たとえば、ログに使用される JSON ファイルのサイズは無制限に増加する可能性があります。つまり、デフォルトでこのファイルのサイズに上限はありません。Docker ソリューションで json-file ログドライバを使用する場合、JSON ログファイルを循環して、JSON ファイルのサイズの制限を検討することをお勧めします。

JSON ログファイルのサイズ制限または json-file ログドライバの他のオプションを指定するには、`docker run` コマンドに 1 つまたは複数のサポートされた json-file ログオプションを指定します。たとえば、以下のコマンドでは is:microPkg イメージを使用する IS_1 という名前の Docker コンテナを開始します。このコンテナには、それぞれの最大サイズが 25 MB である最大 4 つのログファイルを入れることができます。

```
docker run -d --name IS_1 -p 55555:5555
--log-opt max-size=25m --log-opt max-file=4 is:microPkg
```

json-file ログドライバまたは Docker に含まれる 他のログドライバの詳細については、 Docker のマニュアルを参照してください。

コンソールへのサーバログの書き込み

Integration Server への Docker コンテナを開始すると、Integration Server が一部またはすべての監査ログをコンソール (STDOUT) に書き込むよう指示できます。監査ログエントリがコンソールへ書き込まれると、効率的なロギングが提供され、Docker コンテナ内で Integration Server を実行するのに特に役立ちます。複数のコンテナを操作するとき、コンソールからコンテナによって生成されたログエントリを取得できます。これは、各コンテナ内のファイルシステムからログファイルを抽出するのに適しています。

コンソールに書き込まれている各監査ログエントリには、以下が含まれます。

- 識別子としてのロガー名。たとえば、WMERROR はエラーログの識別子であり、WMSESSION はセッションログの識別子です。
- ‖ (二重縦線記号) 文字はフィールド区切り文字としての役割を果たします。watt.server.audit.stdout.fieldDelimiter サーバ設定パラメータを使用して、フィールド区切り文字を変更できます。
- && 記号はレコード区切り文字としての役割を果たします。watt.server.audit.stdout.recordDelimiter サーバ設定パラメータを使用して、レコード区切り文字を変更できます。

コンソールに書き込まれたセッションログエントリの例を以下に示します。

```
jvm 1    | &&WMSESSION||a8d3b287-42bc-4f01-9110-22100f7bd514||?||
a8d3b287-42bc-4f01-9110-22100f7bd514||9f1b2585-db1b-5b47-39f0-d2651aacf8c9||
host:5555||2018-06-26T16:55:19.916Z||2018-06-26T16:55:19.917Z||1||Default||
internal connection||0||0||8798bcf3e47849368c42677bd76bba91||?&&
```

Integration Server が Docker コンテナ内で実行されいる場合、 コンソールに一部またはすべての監査ログを書き込むためには、`docker run` コマンドで Docker コンテナを開始するときに `SAG_IS_AUDIT_STDOUT_LOGGERS` 環境変数を設定します。

`SAG_IS_AUDIT_STDOUT_LOGGERS` を、次のいずれかの値に設定します。

- STDOUT に書き込む監査ロガーのカンマ区切りリストたとえば、「WMSESSION,WMERROR」と入力します。

  監査ロガー名は監査ログファイル名の最初の部分です。監査ログファイルの一覧については、『*webMethods Audit Logging Guide*』を参照してください。

- ロガー名のカンマ区切りリストは、Integration Server Administrator の [**ログ**] > [ログ設定] ページに表示されます。例: セッションロガー、エラーロガー
- `ALL` は、すべてのファイルベースの監査ロガーを STDOUT に書き込みます。
- `NONE` は、どのファイルベースの監査ロガーも STDOUT に書き込まないことを示します。これがデフォルト値です。

注:
`SAG_IS_AUDIT_STDOUT_LOGGERS` の値は大文字と小文字を区別しません。

Integration Server は、STDOUT への監査ログエントリを書き込みを補助出力と見なします。このことは、ロガーが引き続きファイルまたはデータベースの指定された監査ログの記録先に書き込むことを意味します。

Docker コンテナで実行されている Integration Server へのアクセス

Integration Server が Docker コンテナで実行される場合、Integration Server Administrator を使用して Integration Server にアクセスできます。

Docker コンテナで実行されている Integration Server で Integration Server Administrator を使用するには、インターネットブラウザを開き、Integration Server が実行されているコンテナホストマシンの Integration Server ポートにブラウザを指定します。詳細については、Docker のマニュアルを参照してください。

注:
docker run コマンドの実行時に、Docker コンテナをホストするマシンのポートを Docker コンテナの公開ポートに明示的にマップした場合にのみ、Docker コンテナで実行されている Integration Server に対して Integration Server Administrator を使用できます。つまり、コンテナホストマシンのポートは Integration Server のポートにマップする必要があります。Docker コンテナ内の Integration Server 用の Docker イメージの実行の詳細については、1019 ページの「オンプレミス Docker コンテナでの Docker イメージの実行」を参照してください。

Docker コンテナで実行されている Integration Server 用の Integration Server Administrator を起動するには

1. ブラウザを起動します。
2. 以下の形式を使用して、ブラウザにコンテナホストマシンおよびコンテナで公開されている Integration Server ポートを指定します。

   protocol://containerHostMachine:host_port

たとえば、コンテナホストマシンのポート 34678 を Integration Server のプライマリポート 5555 に明示的にマップし、コンテナホストマシン名が machineABC である場合、「http://machineABC: 34678」と入力します。

3. Administrator 特権を持つユーザ名とパスワードで Integration Server にログオンします。

Docker コンテナで Integration Server を実行するときにログファイルと設定ファイルを外部化する

Dockerコンテナで Integration Server を実行する場合、Integration Server では Docker ボリュームを使用して、ホストファイルシステムのマウントされたディレクトリに設定ファイルとログファイルをパーシストすることができます。Docker ボリュームはホストマシンのファイルシステムで Docker コンテナの外側に存在し、直接使用できるようになります。Docker ボリュームを使用して、Integration Server インスタンスと Integration Server プロファイルの logs ディレクトリ、および Docker コンテナで実行されている Integration Server インスタンスの config ディレクトリの内容を外部化できます。

注:
多くのコンテナ展開ソリューションは、コンテナのコンソールログ STDOUT および STDERROR を表示する機能を備えています。Docker コンテナで Integration Server の Docker イメージを実行すると、Integration Server はサーバログを server.log ファイルだけでなくコンソールにも書き込みます。`SAG_IS_AUDIT_STDOUT_LOGGERS` 環境変数を使用して、どのファイルベースの監査ログが STDOUT に書き込むかを示します。詳細については、1022 ページの「コンソールへのサーバログの書き込み」を参照してください。

設定ファイルとログファイルに対してボリュームを使用するには、`docker run` コマンドの実行時に次の環境変数の 1 つまたは複数を指定します。`docker run` コマンドで、環境変数はイメージ名の直前に含めます。

環境変数	説明
`HOST_DIR`	ファイルの書き込み先のホストマシンにある、マウントされたディレクトリへのパス。`docker run` コマンドに `HOST_DIR` パラメータと値を指定すると、Integration Server のログと設定アーティファクトが外部化されます。
`SERVICE_NAME`	Integration Server アーティファクトをパーシストする `HOST_DIR` の下の一意のディレクトリ名。指定すると、logs および config ディレクトリが *<HOST_DIR>/<SERVICE_NAME>* ディレクトリの下に作成され、logs および config ディレクトリのすべてのアーティファクトはそれぞれのディレクトリでパーシストされます。`SERVICE_NAME` を指定しない場合、*<HOST_DIR>* ディレクトリの下に作成される一意のディレクトリ名として Integration Server インスタンス名が使用されます。各 Docker コンテナに一意のディレクトリを指定することをお勧めします。

環境変数	説明
PERSIST_LOGS	「true」に設定すると、Docker コンテナ内で実行されている Integration Server はログファイルを *<HOST_DIR>*/*<SERVICE_NAME>*/logs にパーシストします。SERVICE_NAME が指定されていない場合、*<SERVICE_NAME>* は *<INSTANCE_NAME>* に置き換えられます。Integration Server は *Integration Server_directory* /instances/*<instanceName>*/logs および profiles/IS_*<instanceName>*/logs ディレクトリにあるログを外部化します。「false」に設定すると、Docker コンテナ内で実行されている Integration Server はログファイルを外部化しません。デフォルトは「true」です。
PERSIST_CONFIGS	「true」に設定すると、Docker コンテナ内で実行されている Integration Server は設定ファイルを *<HOST_DIR>*/*<SERVICE_NAME>*/config ディレクトリにパーシストします。SERVICE_NAME が指定されていない場合、*<SERVICE_NAME>* は *<INSTANCE_NAME>* に置き換えられます。「false」に設定すると、Docker コンテナ内で実行されている Integration Server は設定ファイルを外部化しません。デフォルトは「true」です。

例

次の docker run コマンドについて考えます。

```
docker run -d --name IS_Default -p 5555 -p 9999
       -v /opt/myfolder:/opt/myfolder
       -e HOST_DIR=/opt/myfolder -e SERVICE_NAME=demo is:microPkg
```

設定およびログファイルは、 /opt/myfolder/demo/logs に書き込まれたログファイルおよび /opt/myfolder/demo/config に書き込まれた 設定ファイルが /opt/myfolder/demo ディレクトリに書き込まれます。 opt/myfolder/demo/logs ディレクトリには *Integration Server_directory* /instances/*instanceName*/logs ディレクトリおよび profiles/IS_default/logs ディレクトリからのログファイルが含まれます。

注:
上の例で、フォルダが Docker コンテナを実行しているマシンにマウントされておらず、Docker コンテナに表示されない場合のみ、-v /opt/myfolder:/opt/myfolder を指定する必要があります。

Docker イメージを Integration Cloud にプッシュする

オンプレミス Integration Server 用に作成された Docker イメージを Integration Cloud によってホストされた Docker レジストリにプッシュするには、pushImage コマンドを使用します。

注:

Integration Server 用の Docker イメージを webMethods Cloud にプッシュする前に、Integration Server イメージ用のリポジトリがイメージのプッシュ先ステージ用の webMethods Cloud に存在する必要があります。さらに、リポジトリ名は Docker イメージ名のイメージ名部分に対応する必要があります。たとえば、Docker イメージ名「is:microPkg」で、「is」はイメージ名であり、「microPkg」はイメージ名に与えられたタグです。Docker イメージを Integration Cloud にプッシュする前に、「is」という名前のリポジトリが指定されたステージ用の Integration Cloud に存在する必要があります。

Docker イメージを Integration Cloud にプッシュするには

1. 次のディレクトリに移動します。

   Software AG_directory/ Integration Server_directory /docker

2. 次のコマンドを実行します。

   ```
   is_container.sh/bat pushImage required arguments[optional argument]
   ```

引数	説明
`-Duser=` *user ID*	Integration Cloud にアクセスするユーザ名。
`-Dpassword=` *password*	オプション。Integration Cloud にアクセスするパスワード。`pushImage` コマンドでパスワードを指定しなかった場合、後でパスワードの入力が求められます。
`-Dserver=` *URL of* Integration Cloud	Integration Cloud の URL。 例: *subDomain name*.webmethodscloud.com
`-Dsubdomain.name=` *subdomain Name*	Integration Cloud のサブドメイン。 例: *subdomain name*.webmethodscloud.com
`-Dstage.name=`{development|test|prelive|live}	イメージのプッシュ先のステージ。
`-Dimage.name=` *Docker_ image_name*	Integration Server イメージなど、プッシュする Docker イメージの名前。 例: is:microPkg

Integration Cloud での Docker イメージの実行

Integration Cloud における Docker コンテナとリポジトリの管理の詳細と手順については、*webMethods Integration Cloud Help*の Docker コンテナの使用に関する説明を参照してください。

Integration Server の Docker コンテナの停止

`docker stop` コマンドにより、実行されているコンテナが停止します。デフォルトでは、`docker stop` コマンドは、コンテナが停止するのを 10 秒待機してから Docker がコンテナを強制終了します。しかし、デフォルト待機時間の 10 秒は、Integration Server が正常にシャットダウンするには十分ではありません。`docker stop` コマンドを使用する場合、`--time` オプション (`-t`) を使用して、Integration Server が正常にシャットダウンできるよう待機時間を長めに指定します。

Integration Server が正常なシャットダウンを完了するのに必要とする秒数は、Integration Server に設定されたシャットダウンサービスの数、およびシャットダウンサービスの実行が完了するまでにかかる時間によって異なります。Software AG は、シャットダウンサービスのない Integration Server には、待機時間を少なくとも 30 秒に設定することをお勧めします。シャットダウンサービスの数、およびこれらのサービスを実行するのに必要な時間に応じて、待機時間を増やします。

Docker イメージでの設定変数テンプレートの使用

Microservices Runtime ライセンスを備えている Microservices Runtime または Integration Server は、設定変数テンプレートを使用するための機能を提供し、Docker コンテナ内で実行している Microservices Runtime に対して、設定情報を設定します。設定変数テンプレートは、一連のキーと値のペアとしての設定データを含むプロパテファイルです。ここでは、キーの名前には値を指定できるアセットおよび特定のアセットプロパティが反映されています。プロパティの値はテンプレートで設定され、その後、起動時に Microservices Runtime によって適用されます。

設定情報を外部化することにより、Microservices Runtime または Integration Server に作成された単一の Docker イメージを、実稼動サイクルの異なるステージを含む複数の環境で使用できます。たとえば、テストや実稼動のような特定の環境で、異なるテンプレートを使用します。または、すべての環境に同じテンプレートを使用し、環境変数を使用して各環境で設定を変化させます。

設定変数テンプレートの詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。

58 Integration Server Administrator API

- Administrator API の概要 1030
- 認証と承認 1030
- REST の URL 構造 1030
- CRUD 操作 1031
- 管理アクション 1032
- クレデンシャル 1032
- 応答の拡張 1033
- 応答のメディアタイプ 1034
- 要求ペイロードのメディアタイプ 1034
- 使用の開始 1035
- Administrator API へのアクセスの制御 1035
- クライアントエラーのロギング 1036
- 応答本文にスタックトレースを含める 1036
- Integration Server Administrator API 操作 1036

Administrator API の概要

Integration Server が提供する管理 API を使用して、Integration Server の再起動や Integration Serverのアセットの作成、抽出、更新、削除などの管理アクションを実行できます。Administrator API では、REST アーキテクチャ方式を使用します。すべての REST 操作の作業単位はリソースです。Integration Server Administrator API で、リソースはパッケージ、ライセンスキー、サーバ自体などの管理アセットです。

Integration Server Administrator API ではメディアタイプとして JSON を使用します。ただし、Integration Server に対応するメディアタイプのコンテンツハンドラがある限り、API ではクライアントによって要求される他のメディアタイプをサポートします。

Administrator API のすべての要求および応答は OpenAPI 3.0 ドキュメントによって記述されています。このドキュメントは次の方法で取得できます。

- API から `GET /admin/swagger/integrationServer` を使用して要求
- ファイルシステムから: *Integration Server_directory* \instances*instanceName*\packages\WmAdmin\resources\wmadmin_swagger.json

注:
OpenAPI 仕様は元々 Swagger 仕様と呼ばれていました。OpenAPI ドキュメントも Swagger ドキュメントまたはファイル、あるいは単に Swagger と呼ばれることがあります。

Administrator API は WmAdmin パッケージが有効な場合に利用できます。このパッケージは、デフォルトでは有効になっています。WmAdmin パッケージを無効にすると、API は無効になります。

Integration Server Administrator API を使用するには、REST、OpenAPI 3.0、および Integration Server の知識が必要です。

認証と承認

Administrator API では、Integration Server でサポートされているあらゆる形式の認証と承認をサポートします。

REST の URL 構造

Administrator API は、URL パスによりリソースにアクセス可能にします。Administrator API に対するすべての要求は HTTPS または HTTP を使用して行う必要があります。Administrator API では、HTTP の標準的メソッド GET、POST、PUT、PATCH、および DELETE をサポートしています。すべてのリソースがすべての HTTP メソッドをサポートするとは限りません。ただし、すべてのリソースは HEAD と OPTIONS メソッドをサポートします。

Administrator API リソースへの絶対 URL は次のよう構造を持ちます。

`https://host:port/admin/resourceType[/resourceId]`

ここで、

- `https` は転送プロトコルです。HTTP または HTTPS を使用できます。

Software AG では、Integration Server を管理するため、安全性の高い HTTPS プロトコルの使用をお勧めします。

- *host*:*port* は管理対象の Integration Server のホストとポートです。
- admin は Administrator API のディレクティブです。
- *resourceType* はリソースのタイプです (パッケージなど)。
- *resourceId* は特定のリソースを示します。*resourceId* は一部の要求でのみ使用されます。

次に例を示します。

```
HTTP GET http://localhost:port/admin/package
```

すべてのパッケージを抽出します。

```
HTTP GET http://localhost:port/admin/package/WmPublic
```

WmPublic パッケージを抽出します。

Administrator API リソースの相対 URL は /admin で開始した後、リソースのタイプを指定する必要があります。相対 URI は次のような構造を持ちます。

```
/admin/resourceType[/resourceId]
```

次に例を示します。

```
GET /admin/license
```

Integration Server のライセンス情報を返します。

CRUD 操作

CRUD (Create Retrieve Update Delete) 操作は Administrator REST API でサポートされている要求タイプの 1 つです。CRUD 操作はさまざまな HTTP メソッドを使用して実行されます。

次の表に、HTTP メソッドおよび関連する CRUD 操作を示します。

メソッド	操作
GET	1 つ以上のリソースを抽出します。
POST	要求で送信された詳細を使用して新しいリソースを作成します。
PUT	要求で送信された詳細を使用してリソースを置き換えます。
PATCH	要求の本文に指定されているフィールドを置き換えることで、リソースを更新します。
DELETE	リソースを削除します。

注:

すべてのリソースですべてのHTTPメソッドを利用できるわけではありません。特定のリソースで利用できるメソッドの詳細については、Administrator API用のOpenAPI 3.0ドキュメントのAPIに関する説明を確認してください。すべてのリソースはHEADメソッドとOPTIONSメソッドをサポートします。

Administrator APIのPATCH操作はJSONマージパッチ仕様 (https://tools.ietf.org/html/rfc7396) をサポートしています。ただし、Administrator APIは次の点でJSONマージパッチ仕様と異なります。

- 仕様では、クライアントがapplication/merge-patch+jsonメディアタイプを使用することを要求しています。Administrator APIはこのメディアタイプをサポートしますが、application/jsonを使用するPATCH要求も受け入れます。
- 仕様には、「マージパッチのNULL値には、ターゲットの既存の値の削除を示す特別な意味が与えられます。」と記述されています。Administrator APIでは、PATCHを使用して、フィールドの値をNULLに変更できます。その後、更新されたリソースを抽出した場合、応答ペイロードのそのフィールドがNULL値を持つことがあります。Administrator APIにおいて、ペイロードでNULL値があるフィールドは、論理的にはペイロードに存在しないフィールドに等しくなります。

管理アクション

サーバの再起動、パッケージの有効化などの管理アクションはAdministrator APIでサポートされているもう1つの要求タイプです。管理アクションの要求では、POSTメソッドを使用します。URLパスは処理対象のリソースを示します。URLクエリーには、実行するアクションを指定するアクションパラメータがあります。

次に例を示します。

```
POST /admin/package/WarehouseUtils?action=enable
```

WarehouseUtilsパッケージを有効にします。

```
POST /admin/server?action=restart
```

Integration Serverを再起動します。

管理アクションにはクエリーパラメータが追加されることがあります。

次に例を示します。

```
POST /admin/server?action=shutdown&force=true
```

アクティブセッションがある場合でも、Integration Serverを即時にシャットダウンします。

クレデンシャル

Administrator APIはステートレスです。要求が処理されるとき、セッションはIntegration Serverに保存されません。応答でSet-Cookieヘッダーは返されません。管理者クライアントは各要求でクレデンシャルを指定する必要があります。

```
GET /admin/license HTTP/1.1
```

```
Host: host:port
Authorization: Basic <base64-encoded username:password>
Accept: application/json
```

ライセンス情報を抽出します。

応答の拡張

一部の Administrator API 応答には複数のリソースが含まれます。たとえば、要求 `GET /admin/package` の場合、Integration Server はすべてのパッケージを返します。デフォルトで、応答はリンクの配列で構成されます。

```
{
  "packages": [
    {
      "name": "Default",
      "url": "https://localhost:5543/admin/package/Default/"
    },
    {
      "name": "WmAdmin",
      "url": "https://localhost:5543/admin/package/WmAdmin/"
    },
    {
      "name": "WmART",
      "url": "https://localhost:5543/admin/package/WmART/"
    },
    {
      "name": "WmAssetPublisher",
      "url": "https://localhost:5543/admin/package/WmAssetPublisher/"
    },
    {
      "name": "WmCloud",
      "url": "https://localhost:5543/admin/package/WmCloud/"
    },
    etc...
   ]
}
```

リンクの配列を取得する代わりに、 URL にクエリーパラメータ `expand=true` を含めると、Administrator API がパッケージリソース全体を返すよう 要求できます。

```
GET /admin/package?expand=true
```

この場合、応答は次のようになります。

```
{
  "packages": [
    {
      "packageName": "Default",
      "loadok": "1",
      "loaderr": "0",
      "loadwarning": "0",
      "version": "10.5.0.0.5015",
```

```
      "build": "5015",
      "description": "",
      "patch_nums": "",
```

```
      "jvm_version": "1.8",
      "publisher": "Software AG",
      "time": "",
      "listACL": "Default",
      "message": "",
      "code": "1"
    },
```

```
    {
      "packageName": "WmAdmin",
      "loadok": "2",
      "loaderr": "0",
      "loadwarning": "0",
      "startupServices": [
        "wm.admin.controller:init"
      ],
      "version": "10.5.0.0.5015",
      "build": "5015",
      "description": "Administrative API",
      "patch_nums": null,
```

```
      "jvm_version": "1.8",
      "publisher": "Software AG",
      "time": null,
      "listACL": "Administrators",
      "successors": [
        "WmRoot;10.5.0"
```

```
      ],
      "message": "",
      "code": "1"
    },
    etc…
  ]
```

```
}
```

応答のメディアタイプ

API クライアントは、要求に Accept ヘッダーを含めることで、応答のメディアタイプを制御できます。

次の要求は Integration Server からライセンス情報を抽出します。

```
GET /admin/license HTTP/1.1
Host: host:port
Authorization: Basic <base64-encoded username:password>
Accept: text/xml
```

要求ペイロードのメディアタイプ

PUT、PATCH、POST などのペイロードを含む要求には、要求本文のメディアタイプを示す Content-type ヘッダーを含める必要があります。

次の要求では、Integration Server ライセンスキーファイルを更新します。

```
PATCH /admin/license HTTP/1.1
```

```
Host: host:port
Authorization: Basic <base64-encoded username:password>
Content-type: application/json
Accept: application/json

{
    "licenseKeyFile" :  "/user/sysAdmin/licenseKey.xml"
}
```

使用の開始

Administrator API の使用を開始するには、以下の作業を実行する必要があります。

1. WmAdmin パッケージが有効になっていることを確認して、Administrator API が有効になっていることを確認します。
2. CRUD 操作を実行できるユーザに Administrator 特権を割り当て、情報抽出のみできるユーザに読み取り専用権限を割り当てることで、Administrator API へのアクセスを設定します。詳細については、1035 ページの「Administrator API へのアクセスの制御」を参照してください。
3. Administrator API クライアントエラーをエラーログに書き込むかどうか、およびスタックトレースをクライアントへのエラー応答に含めるかどうかを決定して、ロギングとエラーの処理を設定します。詳細については、1035 ページの「Administrator API へのアクセスの制御」および1036 ページの「応答本文にスタックトレースを含める」を参照してください。

Administrator API へのアクセスの制御

Administrator API は 2 つのレベルの権限を提供します。

- Administrators グループのメンバーである管理者は、Administrator API にフルアクセスできます。Administrators グループに属するユーザはすべての CRUD 操作および管理アクションを実行できます。
- GET 要求の発行のみできる読み取り専用管理者。

  読み取り専用管理者が Administrator API の他の HTTP メソッドを使用した場合、サーバは 403 ステータスコードと共にその要求を拒否します。読み取り専用管理者は watt.adminapi.group.readOnly サーバ設定パラメータに割り当てられたユーザグループに属します。

注:
管理者または読み取り専用管理者ではないクライアントが Administrator API 要求を発行した場合、サーバは 403 ステータスコードと共にその要求を拒否します。

読み取り専用管理者グループの作成

読み取り専用管理者グループは GET 要求を使用してのみ Integration Server Administrator API にアクセスできます。

読み取り専用管理者グループを作成するには

1. ユーザグループを作成します。読み取り専用管理者になるユーザをグループに追加します。ユーザグループの作成の詳細については、116 ページの「グループの定義」を参照してください。

2. 拡張設定を使用して、watt.adminapi.group.readOnly の値をユーザグループの名前に設定します。ユーザグループ名は大文字と小文字を区別します。

3. WmAdmin パッケージを再ロードして、変更事項を反映させます。または、Integration Server を再起動します。

クライアントエラーのロギング

Integration Server では、Administrator API の REST リソースの実行中に発生するサーバエラーを常にエラーログに書き込みます。サーバエラーは 500 台の HTTP ステータスコードになるエラーです。ただし、クライアントエラー (400 台の HTTP ステータスコードになるエラー) はデフォルトではエラーログに書き込まれません。クライアントエラーをエラーログに書き込む場合、watt.adminapi.log.clientErrors を「true」に設定します。このプロパティの変更は即座に反映されます。

応答本文にスタックトレースを含める

Administrator API で例外が発生した場合、エラーは HTTP 応答でクライアントに返されます。デフォルトで、応答の本文には、デバッグに役立つ可能性がある例外メッセージとスタックトレースが含まれます。エラー応答でスタックトレースを省略する場合、watt.adminapi.returnExceptions をデフォルトの「false」に設定します。watt.adminapi.returnExceptions を「true」に設定すると、スタックトレースは応答本文に含まれます。このプロパティの変更は即座に反映されます。

Integration Server Administrator API 操作

Integration Server Administrator API には、次の表に示す操作が含まれます。

操作	説明
GET /admin/account/locking	アカウントロックの設定を抽出します。
PATCH /admin/account/locking	アカウントロックの設定を更新します。
GET /admin/account/password/expiration	パスワードの有効期限の設定を抽出します。
PATCH /admin/account/password/expiration	パスワードの有効期限の設定を更新します。
GET /admin/account/password/restriction	パスワード制限の設定を抽出します。

操作	説明
PATCH /admin/account/password/restriction	パスワード制限の設定を更新します。
GET /admin/acl/{name}	アクセスコントロールリストを抽出します。
GET /admin/acl/	すべてのアクセスコントロールリストを抽出します。
POST /admin/acl/	アクセスコントロールリストを作成します。
PATCH /admin/acl/{name}	アクセスコントロールリストを更新します。
PUT /admin/acl/{name}	アクセスコントロールリストを置き換えます。
DELETE /admin/acl/{name}	アクセスコントロールリストを削除します。
GET /admin/cachemanager/	すべてのキャッシュマネージャを抽出します。
GET /admin/cachemanager/{cacheManagerName}	キャッシュマネージャを抽出します。
POST /admin/cachemanager/	キャッシュマネージャを作成します。
POST /admin/cachemanager/{cacheManagerName}	キャッシュマネージャを停止、開始、または再開します。
PATCH /admin/cachemanager/{cacheManagerName}	キャッシュマネージャを更新します。
PUT /admin/cachemanager/{cacheManagerName}	キャッシュマネージャを置き換えます。
DELETE /admin/cachemanager/{cacheManagerName}	キャッシュマネージャおよびキャッシュマネージャに含まれているすべてのキャッシュを削除します。
GET /admin/cachemanager/{cacheManagerName}/cache	要求されたキャッシュマネージャ用のすべてのキャッシュを抽出します。
GET /admin/cachemanager/{cacheManagerName}/cache/{cacheName}	要求されたキャッシュを抽出します。
POST /admin/cachemanager/{cacheManagerName}/cache	キャッシュを作成します。
PATCH /admin/cachemanager/{cacheManagerName}/cache/{cacheName}	キャッシュを更新します。

操作	説明
PUT /admin/cachemanager/{cacheManagerName}/cache/{cacheName}	キャッシュを置き換えます。
DELETE /admin/cachemanager/{cacheManagerName}/cache/{cacheName}	キャッシュを削除し、キャッシュのすべてのデータを廃棄します。
GET /admin/certificate/client	すべてのクライアント認証の詳細を抽出します。
GET /admin/certificate/client/issuer/{issuer}/usage/{usage}/serialNum/{serialNum}	要求されたクライアント認証の詳細を抽出します。
GET /admin/certificate/client	クライアント認証をインポートします。
PATCH /admin/certificate/client/issuer/{issuer}/usage/{usage}/serialNum/{serialNum}	クライアント認証を更新します。
PUT /admin/certificate/client/issuer/{issuer}/usage/{usage}/serialNum/{serialNum}	クライアント認証を置き換えます。
DELETE /admin/certificate/client/issuer/{issuer}/usage/{usage}/serialNum/{serialNum}	クライアント認証を削除します。
GET /admin/certificate/server	サーバ認証の設定を抽出します。
POST /admin/certificate/server/cache	現在のSSL セッションの情報のキャッシュを消去します。
PATCH /admin/certificate/server	サーバ認証を更新します。
PUT /admin/certificate/server	サーバ認証を置き換えます。
GET /admin/csrfguard	CSRF ガードの設定を抽出します。
GET /admin/csrfguard/token	CSRF ガードセキュアトークンを抽出します。
PATCH /admin/csrfguard	CSRF ガードの設定を更新します。
PUT /admin/csrfguard	CSRF ガードの設定を置き換えます。
GET /admin/dashboard/config	ダッシュボード設定を抽出します。
PATCH /admin/dashboard/config	ダッシュボード設定を更新します。
PUT /admin/dashboard/config	ダッシュボード設定を置き換えます。
POST /admin/dashboard/config	ダッシュボード設定で管理アクションを実行します。

操作	説明
GET /admin/enterprisegateway/alertoption	デフォルトの警告設定を抽出します。
PATCH /admin/enterprisegateway/alertoption	デフォルトの警告設定をパッチします。
PUT /admin/enterprisegateway/alertoption	デフォルトの警告設定を更新します。
GET /admin/enterprisegateway/dos	ルール固有のサービス拒否設定を抽出します。
GET /admin/enterprisegateway/dos/deniedip	拒否された IP アドレスのリストを抽出します。
DELETE /admin/enterprisegateway/dos/deniedip/{IP}	拒否された IP アドレスリストから IP アドレスを削除します。
GET /admin/enterprisegateway/dos/dosbyip	IP アドレス別のサービス拒否設定を抽出します。
PATCH /admin/enterprisegateway/dos/dosbyip	IP アドレスのサービス拒否設定を更新します。
PUT /admin/enterprisegateway/dos/dosbyip	IP アドレスのサービス拒否設定を置き換えます。
GET /admin/enterprisegateway/dos/globaldos	サービス拒否のグローバル設定を抽出します。
PATCH /admin/enterprisegateway/dos/globaldos	サービス拒否のグローバル設定を更新します。
PUT /admin/enterprisegateway/dos/globaldos	サービス拒否のグローバル設定を置き換えます。
GET /admin/enterprisegateway/filters	Enterprise Gateway のサポートされたフィルタのリストを抽出します。
GET /admin/enterprisegateway/mobileappprotection	モバイルアプリケーションの保護設定を抽出します。
PATCH /admin/enterprisegateway/mobileappprotection	モバイルアプリケーションの保護設定を更新します。
PUT /admin/enterprisegateway/mobileappprotection	モバイルアプリケーションの保護設定を置き換えます。
GET admin/enterprisegateway/rule	Enterprise Gateway ルールを抽出します。
POST /admin/enterprisegateway/rule	Enterprise Gateway ルールを作成します。

操作	説明
PATCH /admin/enterprisegateway/rule	Enterprise Gateway ルールを更新します。
PUT /admin/enterprisegateway/rule	Enterprise Gateway ルールを置き換えます。
GET /admin/enterprisegateway/rule/alertoption/{ruleName}/ruletype/{ruleType}	ルール固有の警告設定を抽出します。
POST /admin/enterprisegateway/rule/alertoption	ルールの警告設定を更新します。
PUT /admin/enterprisegateway/rule/alertoption	ルールの警告設定を置き換えます。
GET /admin/enterprisegateway/rule/{ruleName}/ruletype/{ruleType}	Enterprise Gateway ルールを抽出します。
DELETE /admin/enterprisegateway/rule/{ruleName}/ruletype/{ruleType}	Enterprise Gateway ルールを削除します。
GET /admin/enterprisegateway/rule/priority/{ruleName}/ruletype/{ruleType}	Enterprise Gateway ルールリストのルールの優先順位を抽出します。
POST /admin/enterprisegateway/rule/priority	ルールリストの Enterprise Gateway ルールの優先順位を更新します。
PUT /admin/enterprisegateway/rule/priority	Enterprise Gateway ルールリストのルールの優先順位を置き換えます。
GET /admin/fileaccesscontrol/	fileAccessControl.cnf からファイルアクセスコントロール設定を抽出します。
PATCH /admin/fileaccesscontrol/	fileAccessControl.cnf のファイルアクセスコントロール設定を更新します。
GET /admin/globalvariable	すべてのグローバル変数を抽出します。
GET /admin/globalvariable/{key}	特定のグローバル変数を抽出します。
POST /admin/globalvariable	グローバル変数を作成します。
PATCH /admin/globalvariable/{key}	グローバル変数を更新します。
PUT /admin/globalvariable/{key}	グローバル変数を更新します。
DELETE /admin/globalvariable/{key}	グローバル変数を削除します。
GET /admin/group/	すべてのグループを抽出します。
GET /admin/group/{groupName}	グループを抽出します。

操作	説明
POST /admin/group/	グループを作成します。
PATCH /admin/group/{groupName}	グループを更新します。
PUT /admin/group/{groupName}	グループを置き換えます。
DELETE /admin/group/{groupName}	グループを削除します。
GET /admin/grpc/channel	すべてのgRPCチャネルを抽出します。
GET /admin/grpc/channel/{channelName}	gRPC チャネルを抽出します。
POST /admin/grpc/channel/{channelName}	gRPCチャネルを有効または無効にします。
PATCH /admin/grpc/channel/{channelName}	gRPC チャネル設定を更新します。
GET /admin/grpc/descriptor	すべての gRPC プロバイダ記述子を抽出します。
GET /admin/grpc/descriptor/{providerDescriptorName}	gRPCプロバイダ記述子を抽出します。
POST /admin/grpc/descriptor/{providerDescriptorName}	gRPCプロバイダ記述子を有効または無効にします。
GET /admin/healthgauge/	すべてのヘルスインジケータを抽出します。
GET /admin/healthgauge/{indicatorName}	特定のヘルスインジケータに関する情報を抽出します。
PATCH /admin/healthgauge/{indicatorName}	ヘルスインジケータを更新します。
POST /admin/healthgauge/{indicatorName}	ヘルスインジケータに対して管理アクションを実行します。
GET /admin/hotdeployment	ホット展開の設定を抽出します。
PATCH /admin/hotdeployment	ホット展開の設定を更新します。
GET /admin/jdbc/function/	すべての JDBC 機能を抽出します。
GET /admin/jdbc/function/{functionAliasName}	JDBC 機能を抽出します。
PATCH /admin/jdbc/function/{functionAliasName}	JDBC 機能を更新します。
POST /admin/jdbc/function/{functionAliasName}	JDBC機能に対して管理アクションを実行します。
GET /admin/jdbc/driver/	すべての JDBC ドライバエイリアスを抽出します。

操作	説明
POST /admin/jdbc/driver/	JDBC ドライバエイリアスを作成します。
GET /admin/jdbc/driver/{driverAliasName}	JDBC ドライバエイリアスを抽出します。
PATCH /admin/jdbc/driver/{driverAliasName}	JDBC ドライバエイリアスを更新します。
PUT /admin/jdbc/driver/{driverAliasName}	JDBC ドライバエイリアスを置き換えます。
DELETE /admin/jdbc/driver/{driverAliasName}	JDBC ドライバエイリアスを削除します。
GET /admin/jdbc/pool/	すべての JDBC プールエイリアスを抽出します。
GET /admin/jdbc/pool/{poolAliasName}	JDBC プールエイリアスを抽出します。
POST /admin/jdbc/pool/	JDBC プールエイリアスを作成します。
PUT /admin/jdbc/pool/{poolAliasName}	JDBC プールエイリアスを置き換えます。
POST /admin/jdbc/pool/{poolAliasName}	JDBC プールエイリアスに対して管理アクションを実行します。
PATCH /admin/jdbc/pool/{poolAliasName}	JDBC プールエイリアスを更新します。
DELETE /admin/jdbc/pool/{poolAliasName}	JDBC プールエイリアスを削除します。
GET /admin/jms	すべての JDBC 接続エイリアスを抽出します。
GET /admin/jms/{jmsAliasName}	特定の JDBC 接続エイリアスを抽出します。
POST /admin/jms	JMS 接続エイリアスを作成します。
POST /admin/jms/{jmsAliasName}	JMS 接続エイリアスを有効または無効にします。
PATCH /admin/jms/{jmsAliasName}	JMS 接続エイリアスを更新します。
PUT /admin/jms/{jmsAliasName}	JMS 接続エイリアスを置き換えます。
DELETE /admin/jms/{jmsAliasName}	JMS 接続エイリアスを削除します。
GET /admin/jmstrigger	すべての JMS トリガーを抽出します。

操作	説明
GET /admin/jmstrigger/{triggerName}	特定の JDBC トリガーを抽出します。
POST /admin/jmstrigger/{triggerName}	すべての JDBC トリガー、特定のタイプ (標準または SOAP-JMS) のすべての JDBC トリガー、または個々の JDBC トリガーを有効化、無効化、または一時停止します。
GET /admin/jndi	すべての JNDI 接続エイリアスを抽出します。
GET /admin/jndi/{jndiAliasName}	特定の JNDI 接続エイリアスの情報を抽出します。
POST /admin/jndi	JNDI 接続エイリアスを作成します。
POST /admin/jndi/{jndiAliasName}	JNDI 接続エイリアスに対して管理アクションを実行します。
PATCH /admin/jndi/{jndiAliasName}	JNDI 接続エイリアスを更新します。
PUT /admin/jndi/{jndiAliasName}	JNDI 接続エイリアスを置き換えます。
DELETE /admin/jndi/{jndiAliasName}	JNDI 接続エイリアスを削除します。
GET /admin/jwt/globalsettings	JWT グローバル設定を抽出します。
PATCH /admin/jwt/globalsettings	JWT グローバル設定を更新します。
GET /admin/jwt/issuer/	JWT 認証のすべての信用のある発行元のリストを抽出します。
GET /admin/jwt/issuer/{issuer}	JWT 認証の特定の信用のある発行元を抽出します。
POST /admin/jwt/issuer/	信用のある発行元を削除します。
PATCH /admin/jwt/issuer/{issuer}	信用のある発行元を更新します。
PUT /admin/jwt/issuer/{issuer}	信用のある発行元を置き換えます。
DELETE /admin/jwt/issuer/{issuer}	信用のある発行元を削除します。
DELETE /admin/jwt/issuer/{issuer}/issuerSkew	発行元スキューマッピングを削除します。
DELETE admin/jwt/issuer/{issuer}/trustStoreAlias/{trustStoreAlias}/certificateAlias/{certificateAlias}	発行元と認証のマッピングを削除します。

操作	説明
GET /admin/kerberos/	ケルベロス設定を抽出します。
PATCH /admin/kerberos/	ケルベロス設定を更新します。
GET /admin/ldap/	LDAP 設定を抽出します。
PATCH /admin/ldap/	LDAP 設定を更新します。
GET /admin/ldap/dir/	すべての LDAP ディレクトリ設定を抽出します。
GET /admin/ldap/dir/{index}	LDAP ディレクトリ設定を抽出します。
POST /admin/ldap/dir/	LDAP ディレクトリ設定を作成します。
PATCH /admin/ldap/dir/{index}	LDAP ディレクトリ設定を更新します。
PUT /admin/ldap/dir/{index}	LDAP ディレクトリ設定を置き換えます。
DELETE /admin/ldap/dir/{index}	LDAP ディレクトリ設定を削除します。
GET /admin/license/	ライセンス情報を抽出します。
POST /admin/license/	Terracotta ライセンスファイルまたは Integration Server ライセンスキーファイルを更新します。
PUT /admin/license/	Terracotta ライセンスファイルまたは Integration Server ライセンスキーファイルを更新します。
PATCH /admin/license/	Terracotta ライセンスファイルまたは Integration Server ライセンスキーファイルを更新します。
GET /admin/log/{logName}	監査ロガーデータを抽出します。
GET /admin/logger/	すべての監査ロガーのリストを返します。
GET /admin/logger/{loggerName}	監査ロガーを抽出します。
POST /admin/logger/{loggerName}	監査ロガーに対して有効化アクションまたは無効化アクションを実行します。
PATCH /admin/logger/{loggerName}	監査ロガーの設定を更新します。
PUT /admin/logger/{loggerName}	監査ロガーの設定を置き換えます。

操作	説明
GET /admin/logger/server	サーバロガーを設定できるすべてのトップレベルコンポーネントを抽出します。
GET /admin/logger/server/{component}	サーバロガー機能コードおよび特定のコンポーネントのログレベルを抽出します。
GET /admin/logger/server/{component}/{facilityCode}	トップレベルコンポーネントのログレベルおよびコンポーネントのすべての機能のログレベルを抽出します。
PATCH /admin/logger/server	Integration Server のデフォルトログレベルを更新します。
PATCH /admin/logger/server/{component}	トップレベルコンポーネントのデフォルトログレベルを更新します。
PATCH /admin/logger/server/{component}/{facilityCode}	コンポーネント内の機能コードのログレベルを更新します。
GET /admin/messaging	すべての webMethods メッセージング接続エイリアスを抽出します。
GET /admin/messaging/{aliasName}	特定の webMethods メッセージング接続エイリアスを抽出します。
POST /admin/messaging/	webMethods メッセージング接続エイリアスを作成します。
POST /admin/messaging/{aliasName}	webMethods メッセージング接続エイリアスを有効または無効にするか、またはデフォルトの webMethods メッセージング接続エイリアスとしてエイリアスを設定します。
PATCH /admin/messaging/{aliasName}	webMethods メッセージング接続エイリアスを更新します。
PUT /admin/messaging/{aliasName}	webMethods メッセージング接続エイリアスを置き換えます。
DELETE /admin/messaging/{aliasName}	webMethods メッセージング接続エイリアスを削除します。
GET /admin/messagingtrigger	すべての webMethods メッセージングトリガーを抽出します。
GET /admin/messagingtrigger/{triggerName}	特定の webMethods メッセージングトリガーを抽出します。

操作	説明
POST /admin/messagingtrigger/{triggerName}	webMethods メッセージングトリガーのメッセージ抽出とメッセージ処理の両方またはいずれか一方をアクティブ化または一時停止します。すべての webMethods メッセージングトリガーを対象とするには、triggerName を「all」に設定します。
GET /admin/mqtt	すべての MQTT 接続エイリアスを抽出します。
GET /admin/mqtt/{connectionAliasName}	MQTT 接続エイリアスを抽出します。
POST /admin/mqtt/	MQTT 接続エイリアスを作成します。
POST /admin/mqtt/{connectionAliasName}	MQTT 接続エイリアスに対して管理アクション (有効化または無効化) を実行します。
PATCH /admin/mqtt/{connectionAliasName}	MQTT 接続エイリアスを更新します。
PUT /admin/mqtt/{connectionAliasName}	MQTT接続エイリアスを置き換えます。
DELETE /admin/mqtt/{connectionAliasName}	MQTT 接続エイリアスを削除します。
GET /admin/mqtttrigger	すべての MQTT トリガーを抽出します。
GET /admin/mqtttrigger/{triggerName}	特定の MQTT トリガーを抽出します。
POST /admin/mqtttrigger/{triggerName}	すべての MQTT トリガーまたは特定の MQTT トリガーを有効化、無効化、または一時停止します。
GET /admin/oauth/client	すべての登録済み OAuth クライアントを抽出します。
GET /admin/oauth/client/{clientId}	指定されたクライアント ID を持つ OAuth クライアントの情報を抽出します。
GET /admin/oauth/client/{name}/version/{version}	指定されたクライアント名およびバージョンを持つ OAuth クライアントの情報を抽出します。
POST /admin/oauth/client	OAuth クライアントを作成します。
PATCH /admin/oauth/client/{clientId}	指定されたクライアント ID を持つ OAuth クライアントのクライアント情報を更新します。

操作	説明
PUT /admin/oauth/client/{clientId}	指定されたクライアント ID を持つ OAuth クライアントのクライアント情報を置き換えます。
DELETE /admin/oauth/client/{clientId}	OAuth クライアントを削除します。
GET /admin/oauth/externalserver	すべての外部認証サーバを抽出します。
GET /admin/oauth/externalserver/{name}	外部認証サーバを抽出します。
POST /admin/oauth/externalserver/	外部認証サーバを作成します。
PATCH /admin/oauth/externalserver/{name}	外部認証サーバを更新します。
PUT /admin/oauth/externalserver/{name}	外部認証サーバを置き換えます。
DELETE /admin/oauth/externalserver/{name}	外部認証サーバを削除します。
GET /admin/oauth/globalsetting	グローバル OAuth 設定を抽出します。
PATCH /admin/oauth/globalsetting/	グローバル OAuth 設定を更新します。
PUT /admin/oauth/globalsetting/	グローバル OAuth 設定を置き換えます。
GET /admin/oauth/scope	スコープのリストを抽出します。
GET /admin/oauth/scope/{name}	スコープを抽出します。
POST /admin/oauth/scope	スコープを作成します。
PATCH /admin/oauth/scope/{name}	スコープを更新します。
PUT /admin/oauth/scope/{name}	スコープを置き換えます。
DELETE /admin/oauth/scope/{name}	スコープを削除します。
GET /admin/oauth/token	アクティブなアクセストークンおよびリフレッシュトークンのリストを抽出します。
GET /admin/oauth/token/accesstoken/{accessTokenId}	アクセストークンを抽出します。
GET /admin/oauth/token/refreshtoken/{refreshTokenId}	リフレッシュトークンを抽出します。
DELETE /admin/oauth/token/accesstoken/{accessTokenId}	アクセストークンを削除します。
DELETE /admin/oauth/token/refreshtoken/{refreshTokenId}	リフレッシュトークンを削除します。

操作	説明
PUT /admin/outboundPassword/masterPassword	マスターパスワードを更新します。
POST /admin/outboundPassword/masterPassword/reset	保存された送信パスワードおよびマスターパスワードを含む、保存されたすべてのパスワードをリセットします。
GET /admin/outboundPassword/masterPassword/expiration	マスターパスワードの有効期限プロパティを抽出します。
PUT /admin/outboundPassword/masterPassword/expiration	マスターパスワードの有効期間を抽出します。
GET /admin/package	すべてのパッケージを抽出します。
GET /admin/package/{packageName}	特定のパッケージのパッケージ情報を取得します。
POST /admin/package/{packageName}	インストール済みパッケージに対して管理アクションを実行します。アクションは無効化、有効化、再ロード、アクティブ化、アーカイブ、または回復です。
DELETE /admin/package/{packageName}	パッケージを削除します。
GET /admin/port	Integration Server またはパッケージのすべてのポートを抽出します。この操作では、拡張パラメータをサポートしていません。
GET /admin/port/diagnostic	Integration Server の診断ポートへのリンクを抽出します。
GET /admin/port/primary	Integration Server のプライマリポートへのリンクを抽出します。
POST /admin/port	Integration Server のプライマリポートを設定します。
GET /admin/port/ipaccess	グローバル IP アクセスの設定を抽出します。
PUT /admin/port/ipaccess	グローバル IP アクセスの設定を置き換えます。
GET /admin/port/email	すべての電子メールポートを抽出します。
GET /admin/port/email/{alias}	電子メールポートを抽出します。

操作	説明
POST /admin/port/email	電子メールポートを作成します。
POST /admin/port/email/{alias}	電子メールポートを有効または無効にします。
PATCH /admin/port/email/{alias}	電子メールポートを更新します。
PUT /admin/port/email/{alias}	電子メールポートを置き換えます。
DELETE /admin/port/email/{alias}	電子メールポートを削除します。
GET /admin/port/email/{alias}/ipaccess	電子メールポートの IP アクセス設定を抽出します。
PUT /admin/port/email/{alias}/ipaccess	電子メールポートの IP アクセス設定を置き換えます。
DELETE /admin/port/email/{alias}/ipaccess	電子メールポートの IP アクセス設定を削除します。削除後、ポートではグローバル IP アクセス設定を使用します。
GET /admin/port/email/{alias}/resourceaccess	電子メールポートのリソースアクセス設定を抽出します。
PUT /admin/port/email/{alias}/resourceaccess	電子メールポートのリソースアクセス設定を置き換えます。
GET /admin/port/enterprisegateway	すべての Enterprise Gateway ポートを抽出します。
GET /admin/port/enterprisegateway/{alias}	Enterprise Gateway ポートを抽出します。
POST /admin/port/enterprisegateway/	Enterprise Gateway ポートを作成します。
POST /admin/port/enterprisegateway/{alias}	Enterprise Gateway ポートを有効または無効にします。
PATCH /admin/port/enterprisegateway/{alias}	Enterprise Gateway ポートを更新します。
PUT /admin/port/enterprisegateway/{alias}	Enterprise Gateway ポートを置き換えます。
DELETE /admin/port/enterprisegateway/{alias}	Enterprise Gateway ポートを削除します。
GET /admin/port/enterprisegateway/{alias}/ipaccess	Enterprise Gateway ポートの IP アクセス設定を抽出します。

操作	説明
PUT /admin/port/enterprisegateway/{alias}/ipaccess	Enterprise Gateway ポートの IP アクセス設定を置き換えます。
GET /admin/port/ftp/	すべての FTP ポートを抽出します。
GET /admin/port/ftp/{alias}	FTP ポートを抽出します。
POST /admin/port/ftp/	FTP ポートを作成します。
POST /admin/port/ftp/{alias}	FTP ポートを有効または無効にします。
PATCH /admin/port/ftp/{alias}	FTP ポートを更新します。
PUT /admin/port/ftp/{alias}	FTP ポートを置き換えます。
DELETE /admin/port/ftp/{alias}	FTP ポートを削除します。
GET /admin/port/ftp/{alias}/ipaccess	FTP ポートの IP アクセス設定を抽出します。
PUT /admin/port/ftp/{alias}/ipaccess	FTP ポートの IP アクセス設定を置き換えます。
DELETE /admin/port/ftp/{alias}/ipaccess	FTP ポートの IP アクセス設定を削除します。削除後、ポートではグローバル IP アクセス設定を使用します。
GET /admin/port/ftp/{alias}/resourceaccess	FTP ポートのリソースアクセス設定を抽出します。
PUT /admin/port/ftp/{alias}/resourceaccess	FTP ポートのリソースアクセス設定を置き換えます。
GET /admin/port/ftps/	すべての FTPS ポートを抽出します。
GET /admin/port/ftps/{alias}	FTPS ポートを抽出します。
POST /admin/port/ftps/	FTPS ポートを作成します。
POST /admin/port/ftps/{alias}	FTPS ポートを有効または無効にします。
PATCH /admin/port/ftps/{alias}	FTPS ポートを更新します。
PUT /admin/port/ftps/{alias}	FTPS ポートを置き換えます。
DELETE /admin/port/ftps/{alias}	FTPS ポートを削除します。
GET /admin/port/http/	すべての HTTP ポートを抽出します。
GET /admin/port/http/{alias}	HTTP ポートを抽出します。

操作	説明
POST /admin/port/http/	HTTP ポートを作成します。
PATCH /admin/port/http/{alias}	HTTP ポートを更新します。
PUT /admin/port/http/{alias}	HTTP ポートを置き換えます。
DELETE /admin/port/http/{alias}	HTTP ポートを削除します。
POST /admin/port/http/{alias}	HTTP ポートを有効化、無効化、一時停止、または再開します。
GET /admin/port/http/{alias}/ipaccess	HTTP ポートの IP アクセス設定を抽出します。
PUT /admin/port/http/{alias}/ipaccess	HTTP ポートの IP アクセス設定を置き換えます。
DELETE /admin/port/http/{alias}/ipaccess	HTTP ポートの IP アクセス設定を削除します。削除後、ポートではグローバル IP アクセス設定を使用します。
GET /admin/port/http/{alias}/resourceaccess	HTTP ポートのリソースアクセス設定を抽出します。
PUT /admin/port/http/{alias}/resourceaccess	HTTP ポートのリソースアクセス設定を置き換えます。
GET /admin/port/https/	すべての HTTPS ポートを抽出します。
GET /admin/port/https/{alias}	HTTPS ポートを抽出します。
POST /admin/port/https/	HTTPS ポートを作成します。
PATCH /admin/port/https/{alias}	HTTPS ポートを更新します。
PUT /admin/port/https/{alias}	HTTPS ポートを置き換えます。
DELETE /admin/port/https/{alias}	HTTPS ポートを削除します。
POST /admin/port/https/{alias}	HTTPS ポートを有効化、無効化、一時停止、または再開します。
GET /admin/port/https/{alias}/ipaccess	HTTPS ポートの IP アクセス設定を抽出します。
PUT /admin/port/https/{alias}/ipaccess	HTTPS ポートの IP アクセス設定を置き換えます。
DELETE /admin/port/https/{alias}/ipaccess	HTTPS ポートの IP アクセス設定を削除します。削除後、ポートではグローバル IP アクセス設定を使用します。

操作	説明
GET /admin/port/https/{alias}/resourceaccess	HTTPS ポートのリソースアクセス設定を抽出します。
PUT admin/port/https/{alias}/resourceaccess	HTTPS ポートのリソースアクセス設定を置き換えます。
GET /admin/port/internalserver/	すべての内部サーバポートを抽出します。
GET /admin/port/internalserver/{alias}	内部サーバポートを抽出します。
POST /admin/port/internalserver	内部サーバポートを作成します。
POST /admin/port/internalserver/{alias}	内部サーバポートを有効または無効にします。
PATCH /admin/port/internalserver/{alias}	内部サーバポートを更新します。
PUT /admin/port/internalserver/{alias}	内部サーバポートを置き換えます。
DELETE /admin/port/internalserver/{alias}	内部サーバポートを削除します。
GET /admin/port/internalserver/{alias}/resourceaccess	内部サーバポートのリソースアクセス設定を抽出します。
PUT /admin/port/internalserver/{alias}/resourceaccess	内部サーバポートのリソースアクセス設定を置き換えます。
GET /admin/port/websocket	すべての WebSocket ポートおよび WebSocketSecure ポートを抽出します。
GET /admin/port/websocket/{alias}	WebSocket ポートまたは WebSocketSecure ポートを抽出します。
POST /admin/port/websocket	WebSocket ポートまたは WebSocketSecure ポートを作成します。
POST /admin/port/websocket/{alias}	WebSocket ポートまたは WebSocketSecure ポートを有効または無効にします。
PATCH /admin/port/websocket/{alias}	WebSocket ポートまたは WebSocketSecure ポートを更新します。
PUT /admin/port/websocket/{alias}	WebSocket ポートまたは WebSocketSecure ポートを置き換えます。

操作	説明
DELETE /admin/port/websocket/{alias}	WebSocket ポートまたは WebSocketSecure ポートを削除します。
GET /admin/port/websocket/{alias}/ipaccess	WebSocket ポートまたは WebSocketSecure ポートの IP アクセス設定を抽出します。
PUT /admin/port/websocket/{alias}/ipaccess	WebSocket ポートまたは WebSocketSecure ポートの IP アクセス設定を置き換えます。
DELETE /admin/port/websocket/{alias}/ipaccess	WebSocket ポートまたは WebSocketSecure ポートの IP アクセス設定を削除します。削除後、ポートではグローバル IP アクセス設定を使用します。
GET /admin/port/websocket/{alias}/resourceaccess	WebSocket ポートまたは WebSocketSecure ポートのリソースアクセス設定を抽出します。
PUT /admin/port/websocket/{alias}/resourceaccess	WebSocket ポートまたは WebSocketSecure ポートのリソースアクセス設定を置き換えます。
GET /admin/proxy/	すべてのプロキシサーバエイリアスを抽出します。
GET /admin/proxy/{proxyAlias}	プロキシサーバエイリアスを抽出します。
POST /admin/proxy/	プロキシサーバエイリアスを作成します。
POST /admin/proxy/{proxyAlias}}	プロキシサーバエイリアスに対して管理アクションを実行します。
PATCH /admin/proxy/{proxyAlias}	プロキシサーバエイリアスを更新します。
PUT /admin/proxy/{proxyAlias}	プロキシサーバエイリアスを置き換えます。
DELETE /admin/proxy/{proxyAlias}	プロキシサーバエイリアスを削除します。
GET /admin/quiesce	現在のサーバモード (アクティブまたは休止) および休止ポートを抽出します。

操作	説明
POST /admin/quiesce	Integration Server をアクティブモードから休止モードにするか、休止モードからアクティブモードにします。
POST /admin/quiesce/port	指定されたポートを休止ポートとして設定します。
GET /admin/saml	すべての SAML トークン発行元を抽出します。
GET /admin/saml/{issuer}	要求された SAML トークン発行元を抽出します。
POST /admin/saml/	SAML トークン発行元を作成します。
PUT /admin/saml/{current_issuer}	置き換え操作の前に、current_issuer がトークン発行元の名前である SAML トークン発行元を置き換えます。
PATCH /admin/saml/{current_issuer}	更新前に、current_issuer がトークン発行元の名前である SAML トークン発行元を更新します。
DELETE /admin/saml/{issuer}	SAML トークン発行元を削除します。
GET /admin/scheduledtask/	スケジュールされているすべてのユーザタスクを抽出します。
GET /admin/scheduledtask/{taskId}	スケジュールされているユーザタスクを抽出します。
POST /admin/scheduledtask/	スケジュールされているユーザタスクを作成します。
POST /admin/scheduledtask/{taskId}	スケジュールされているユーザタスクを一時停止、開始、またはキャンセルします。
PATCH /admin/scheduledtask/{taskId}	スケジュールされているユーザタスクを更新します。
PUT /admin/scheduledtask/{taskId}	スケジュールされているユーザタスクを置き換えます。
DELETE admin/scheduledtask/{taskId}	スケジュールされているユーザタスクを削除します。
GET /admin/scheduler	スケジューラの現在の状態 (実行中または一時停止) およびスケジュールされているタスクのリストを抽出します。

操作	説明
POST /admin/scheduler	スケジューラを再開または一時停止します。
GET /admin/server	Integration Server に関する情報を抽出します。
POST /admin/server	Integration Server に対して管理アクションを実行します。アクションには停止、再起動、restartQuiesce、休止、exitQuiesce があります。
GET /admin/server/diagnostics	診断アーカイブを ZIP 添付として返します。
GET /admin/server/httpout	送信 HTTP 設定を抽出します。
PATCH /admin/server/httpout	送信 HTTP 設定を更新します。
GET /admin/server/memory	Integration Server の利用可能なメモリ情報を抽出します。
GET /admin/server/notification	電子メール通知の設定を抽出します。
PATCH /admin/server/notification	電子メール通知の設定を更新します。
GET /admin/server/session	セッション設定を抽出します。
PATCH /admin/server/session	セッション設定を更新します。
GET /admin/server/setting	Integration Server のすべての拡張設定の値を抽出します。
GET /admin/server/setting/{settingName}	指定された拡張設定の値を抽出します。
POST /admin/server/setting	指定された拡張設定をデフォルト値にリセットします。
PATCH /admin/server/setting/{settingName}	拡張設定を更新します。
PATCH /admin/server/setting/	複数の拡張設定を更新します。
PUT /admin/server/setting/	複数の拡張設定を更新します。
GET /admin/server/sso	MWS シングルサインオンの設定を抽出します。
PATCH /admin/server/sso	MWS シングルサインオンの設定を更新します。
GET /admin/server/threadpool	サーバスレッドプールの設定を抽出します。

操作	説明
PATCH /admin/server/threadpool	サーバスレッドプールの設定を更新します。
GET /admin/server/updates	Integration Server にインストールされた修正に関する情報を抽出します。
GET /admin/sftpserver	すべての SFTP サーバエイリアスを抽出します。
GET /admin/sftpserver/{alias}	特定の SFTP サーバエイリアスを抽出します。
POST /admin/sftpserver	SFTP サーバエイリアスを作成します。
PATCH /admin/sftpserver/{alias}	SFTP サーバエイリアスを更新します。
PUT /admin/sftpserver/{alias}	SFTP サーバエイリアスを置き換えます。
DELETE /admin/sftpserver/{alias}	SFTP サーバエイリアスを削除します。
GET /admin/sftpuser	すべての SFTP ユーザエイリアスを抽出します。
GET /admin/sftpuser/{alias}	特定の SFTP ユーザエイリアスを抽出します。
POST /admin/sftpuser	SFTP ユーザエイリアスを作成します。
PATCH /admin/sftpuser/{alias}	SFTP ユーザエイリアスを更新します。
PUT /admin/sftpuser/{alias}	SFTP ユーザエイリアスを置き換えます。
DELETE /admin/sftpuser/{alias}	SFTP ユーザエイリアスを削除します。
GET /admin/swagger/{productName}	製品の API の Swagger ドキュメントを抽出します。
GET /admin/swagger	Integration Server のすべての管理 API の Swagger ドキュメントを抽出します。 Integration Server が Administrator API を備える唯一のインストール済み製品である場合、Integration Server の Swagger ドキュメントが返されます (GET /admin/swagger/integrationServer と同じ)。

操作	説明
GET /admin/user/	すべてのユーザアカウントを抽出します。
GET /admin/user/{username}	ユーザアカウントを抽出します。
POST /admin/user/	ユーザアカウントを作成します。
POST /admin/user/{username}	ユーザアカウントを有効または無効にします。
PATCH /admin/user/{username}	ユーザアカウントを更新します。
PUT /admin/user/{username}	ユーザアカウントを置き換えます。
DELETE /admin/user/{username}	ユーザアカウントを削除します。
GET admin/urlalias	すべての URL エイリアスを抽出します。
GET /admin/urlalias/{urlAliasName}	要求された URL エイリアスを抽出します。
POST /admin/urlalias/	URL エイリアスを作成します。
PATCH /admin/urlalias/{urlAliasName}	URL エイリアスを更新します。
PUT /admin/urlalias/{urlAliasName}	URL エイリアスを置き換えます。
DELETE /admin/urlalias/{urlAliasName}	URL エイリアスを削除します。

59 JSON 出力の制御

- JSON 応答の日付エンコーディングの制御 1060
- JSON 出力の書式設定して読みやすくする 1060

Integration Server は、Accept: application/json ヘッダーを含む要求への応答で JSON コンテンツのエンコードまたは書式設定を指定するグローバル設定を備えています。また、Integration Server は、クライアントが JSON コンテンツのエンコードまたは書式設定を指定するため、クライアントが要求 URL に含めることができるクエリーパラメータも備えています。要求 URL のクエリーパラメータ値はグローバル設定を上書きします。

JSON 応答の日付エンコーディングの制御

JSON は日付データタイプを定義していないため、Integration Server では watt.server.json.encodeDateAs サーバ設定を使用して、java.util.Date のインスタンスを応答でどのようにエンコードするかを決定します。応答のパイプラインに java.util.Date のインスタンスである日付オブジェクトが含まれる場合、Integration Server ではサーバ設定パラメータの値を使用して、日付オブジェクトをどのようにエンコードするかを決定します。

Integration Server への要求には、JSON 応答の日付エンコーディングを指定するクエリーパラメータを含めることができます。要求 URL の `jsonDateEncoding` クエリーパラメータは watt.server.json.encodeDateAs サーバ設定パラメータの値を上書きします。java.util.Date のインスタンスである日付オブジェクトが応答に使用されるパイプラインに存在する場合、要求 URL の `jsonDateEncoding` クエリーパラメータは、応答で java.util.date フィールドをエンコードする方法を Integration Server に指示します。

`jsonDateEncoding` クエリーパラメータは以下のいずれか 1 つに設定できます。

- NULL: パターン `EEE MMM dd HH:mm:ss zzz yyyy` を使用して、java.util.Date のインスタンスをエンコードします。
- `long`: java.util.Data インスタンスをタイムスタンプとしてエンコードします (具体的には、1970 年 1 月 1 日 00:00:00 以降のミリ秒数)。日付は JSON 数値としてエンコードされます。
- `ISO8601`: 標準 ISO 形式の `YYYY-MM-DD'T'HH:mm:ss.sssZ` で、java.utilData インスタンスをエンコードします。日付は JSON ストリングとしてエンコードされます。
- カスタム形式: 日付は指定されたパターンでエンコードされます。指定するパターンは Oracle Java API のマニュアルの java.text.SimpleDateFormat クラスに記述された日時パターンに準拠する必要があります。日付は JSON ストリングとしてエンコードされます。

このクエリーパラメータは admin、invoke、rest、および restv2 のディレクティブで有効です。

例: `GET /invoke/pub.date/getCurrentDate?jsonDateEncoding=YYYY/MM/DD%20HH:mm:ss`

Integration Server では、ヘッダーに Content-type: Application/json が追加されている場合にのみ、このクエリーパラメータを使用します。

JSON 出力の書式設定して読みやすくする

Integration Server には、応答の JSON コンテンツを読みやすくするプリティプリントオプションが含まれています。watt.server.json.prettyPrint を「true」に設定すると、Integration Server は復帰 (CR) およびインデントを使用して JSON コンテンツを書式設定し、出力を読みやすくします。

サービスの呼び出しに使用される URL に `jsonPrettyPrint=true` または `jsonPrettyPrint=false` のクエリーパラメータを指定することで、個々の要求で watt.server.json.prettyPrint 値を上書きできます。サービスを呼び出す URL に `jsonPrettyPrint=true` を含めると、Integration Server は復帰 (CR) およびインデントを使用して応答を書式設定します。

このクエリーパラメータは invoke、admin、rest、および restv2 のディレクティブで有効です。

Integration Server では、ヘッダーに Content-type: Application/json が追加されている場合にのみ、このクエリーパラメータを使用します。

60 Command Central を使用した Integration Server の管理

- Command Central の概要 1064
- Integration Server インスタンス管理 1064
- Command Central を使用した Integration Server Administrator へのアクセス 1071
- Integration Server インスタンスの監視 1073
- Integration Server の設定タイプ 1075
- Integration Server のライフサイクルアクション 1082
- Integration Server でサポートしているコマンド 1082
- UNIX シェルスクリプトを使用した管理製品の接続クレデンシャルの変更 1083
- Command Central コマンドラインツールのアップグレード 1084

この章では、Command Central 内での Integration Server インスタンスの作成、管理、監視方法を説明します。Command Central でサポートされている Integration Server 設定タイプの詳細については、このマニュアルの対応する箇所を参照してください。

例えば、Command Central でサポートされている Integration Server の SFTP サーバエイリアス設定タイプは474 ページの「SFTP サーバエイリアスの作成」で説明されています。同様に、SFTP ユーザエイリアス設定タイプは477 ページの「SFTP ユーザエイリアスの作成」で説明されています。

注:
Command Central は Software AG Platform Manager を使用して、Integration Server と通信します。Software AG Platform Manager は Integration Server とのアクティブな接続を保持し、60 秒の定期的な間隔で Integration Server インスタンスに ping を実行します。

Command Central の概要

Software AG Command Central は、リリースマネージャ、インフラストラクチャエンジニア、システム管理者、およびオペレータが使用し、管理タスクを 1 か所から実行できるようにするツールです。Command Central は次の設定、管理、および監視タスクを助けることができます。

- インフラストラクチャエンジニアは、どの製品と修正がどこにインストールされているかが一見してわかります。また、エンジニアは、不整合を発見するために、インストール環境を簡単に比較することもできます。
- システム管理者は、単一の Web UI、コマンドラインツールまたは API を使用して環境を設定できるため、保守を実行する場合の作業とリスクを最小化できます。
- リリースマネージャは、コマンドラインスクリプティングを使用して、複数のサーバに対する変更を準備および展開でき、これにより、ライフサイクル管理がより簡単かつ安全になります。
- オペレータは、サーバ状態を監視できると同時に、単一の場所からサーバを起動および停止できます。また、計画外のシステム停止が発生した場合にアラートが送信されるように設定することもできます。

Software AG Command Central の詳細については、Command Central のマニュアルを参照してください。

Integration Server インスタンス管理

Integration Server インスタンスは Command Central インスタンス管理コマンドおよび Command Central Web ユーザインタフェースを使用して管理できます。

注:
Command Central は Microservices Runtime の管理に使用できません。

Command Central Web ユーザインタフェースを使用したインスタンスの作成

Command Central Web ユーザインタフェースを使用して、Integration Server インスタンスを作成できます。

インスタンスを作成するには

1. [**Environments**] パネルで、新しい製品インスタンスを作成する環境を選択します。
2. [**Installations**] タブをクリックします。
3. 新しいインスタンスを作成するインストールを選択します。
4. [**Instances**] タブをクリックします。
5. をクリックして Integration Server を選択します。
6. 次の情報を入力し、[**Next**] をクリックします。

フィールド	指定内容
[Instance name]	新規インスタンスの名前。
[IP Address]	オプション。Integration Server のこのインスタンスでポートをバインドする先のデフォルト IP アドレス。マシンに複数の IP アドレスがあり、そのマシンで Integration Server の異なるインスタンス間で同じポート番号を使用する場合、バインドアドレスを指定します。この場合、各インスタンスには異なるバインドアドレスを指定する必要があります。デフォルトのバインドアドレスを指定しない場合、ポートはすべてのネットワークインタフェースを受信待機し、複数のインタスタンスで同じポート番号が使用されません。 **注:** Integration Server では、指定された *default_bind_address* 値を watt.server.inetaddress 設定プロパティの値として使用します。
[License key file]	オプション。Integration Server ライセンスファイルの場所。
[Register Windows service for automatic startup]	オプション。インスタンスにサービスを登録するかどうか。デフォルトは「`false`」です。
[Administrator Password]	オプション。このインスタンスの Administrator ユーザアカウントのパスワード。Administrator のパスワードをここで指定

フィールド		指定内容
		しないと、パスワードは Integration Server のインストール時に指定されたデフォルト値になります。
[Change Administrator password on first login]		オプション。ログインに最初に成功した後、Administrator ユーザアカウントのパスワードを変更する必要があるかどうか。最初のログイン時にパスワードの変更が必要な場合、「Yes」に設定します。値を指定しない場合、新しいインスタンスでは Integration Server のインストール時に設定されたログイン時のパスワード変更設定がデフォルトになります。
[Database]	[Database type]	新しいサーバインスタンスが使用するデータベースのタイプ。次のいずれかのデータベースを指定します。 ■ `sql`: Microsoft SQL Server ■ `oracle`: Oracle ■ `db2`: DB2 これらの値は、大文字と小文字を区別しません。デフォルトは、組み込みの IS 内部データベースです。
	[JDBC URL]	データベースの接続 URL。
	[Database user]	Integration Server データベースに割り当てられたユーザ名。
	[Password]	Integration Server データベースユーザのパスワード。
	[Connection name]	データベースの接続名 (エイリアス名)。
[Ports]	[Primary port]	オプション。 新しいインスタンスのデフォルト HTTP ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は `5555` です。
	[Diagnostic port]	オプション。新しいインスタンスのデフォルト診断ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は `9999` です。

フィールド	指定内容	
	[JMX port]	オプション。新しいインスタンスのデフォルト JMX ポート番号。ポート番号は、各 Integration Server インスタンスで一意である必要があります。デフォルト値は 8075 です。
[Packages to deploy to this instance]	サーバインスタンスに追加するパッケージ。 [Available] パネルからパッケージを選択し、 をクリックして、[Selected] パネルへそのパッケージを移動します。[Available] パネルに表示されたすべてのパッケージを追加するには、 をクリックします。	

7. [Finish] をクリックします。

Command Central Web ユーザインタフェースを使用したインスタンスの更新

Command Central Web ユーザインタフェースを使用して、Integration Server インスタンスを更新できます。

インスタンスを更新するには

1. [Environments] パネルで、新しい製品インスタンスを作成する環境を選択します。
2. [Installations] タブをクリックします。
3. インスタンスがあるインストールを選択します。
4. [Instances] タブをクリックします。
5. Integration Server プロファイルを選択して、 をクリックします。
6. [Update Instance] を選択します。
7. 次の情報を入力し、[Next] をクリックします。
   - パッケージを追加するには、[Packages to add to this instance] をクリックし、[Available] パネルからパッケージを選択し、 をクリックしてパッケージを [Selected] パネルに移動します。[Available] パネルに表示されたすべてのパッケージを追加するには、 をクリックし、追加するパッケージを選択します。

     注:

[**Packages to add to this instance**] ダイアログボックスに、インスタンスに既にインストールされているパッケージは表示されません。

- データベースプロパティを更新するには、[**Database**] タブをクリックし、詳細を入力します。

注:

データベースプロパティの詳細については、1065 ページの「Command Central Web ユーザインタフェースを使用したインスタンスの作成」を参照してください。

8. [**Finish**] をクリックします。

Command Central Web ユーザインタフェースを使用したインスタンスの削除

Command Central Web ユーザインタフェースを使用して、Integration Server インスタンスを削除できます。

インスタンスを削除するには

1. [**Environments**] パネルで、新しい製品インスタンスを作成する環境を選択します。
2. [**Installations**] タブをクリックします。
3. 削除するインスタンスがあるインストールを選択します。
4. [**Instances**] タブをクリックします。
5. Integration Server プロファイルから Integration Server インスタンスを選択し、 をクリックします。
6. [**OK**] をクリックします。

Command Central インスタンス管理コマンドを使用したインスタンスの管理

以下の表に、Command Central インスタンス管理コマンドを使用して Integration Server インスタンスを管理するときに含める必要がある必須パラメータを示します。

コマンド	パラメータ	説明
`sagcc create instances`	`instance.name=`*`name`*	必須です。新しい Integration Server インスタンスの名前。

コマンド	パラメータ	説明
	`primary.port=`*`port`*	必須。新しい Integration Server インスタンスのメイン受信待機ポート。
	`diagnostic.port=`*`port`*	必須。新しい Integration Server インスタンスの診断ポート。
	`jmx.port=`*`port`*	オプション。新しい Integration Server インスタンスの JMX ポート。
	`install.service=false|true`	オプション。Integration Server をアプリケーションとしてインストールするか、サービスとしてインストールするかを指定します。有効な値は次のとおりです。 ■ `true` - サービスとしてインストールします。 ■ `false` - アプリケーションとしてインストールします (デフォルト)。
	`license.file=`*`location_of_license_file`*	オプション。Integration Server ライセンスファイルの場所。 スクリプトはライセンスファイルを指定された場所から次の場所にコピーします。 IntegrationServer/instances/*instance_name*/config/licenseKey.xml このパラメータはオプションですが、ライセンスファイルの場所が指定されない場合、Integration Server インスタンスは 30 分後にシャットダウンします。 **注:** 場所名にスペースが含まれている場合、場所名を引用符 (" ") で囲む必要があります。
	`db.alias=` *`database_ alias_name`*	オプション。データベースのエイリアス名。デフォルト値は「`JDBC_POOL_ALIAS`」です。
	`db.type=`*`database_type`*	必須。新しいサーバインスタンスが使用するデータベースのタイプ。次のいずれかのデータベースを指定します。

コマンド	パラメータ	説明
		■ `sql`: Microsoft SQL Server ■ `oracle`: Oracle ■ `db2`: DB2 これらの値は、大文字と小文字を区別しません。デフォルトは、組み込みの IS 内部データベースです。
	`db.username=`*`database_ username`*	必須。Integration Server データベースに割り当てられたユーザ名。
	`db.password=`*`database_ password`*	必須。Integration Server データベースのパスワード。
	`db.url=`*`database_URL`*	必須。データベースの接続 URL。

Command Central での実行時の例

- インストールされた Integration Server に対し、エイリアス名 productionNode2 のインストールでインスタンス名 is-instance2、診断ポート 8083、プライマリポート 8081 の新しいインスタンスを作成するには

```
sagcc create instances productionNode2 integrationServer
instance.name=is-instance2 diagnostic.port=8083
primary.port=8081
db.alias=sql_server db.type=sqlserver
db.username=sa db.password=password@1234
db.url="jdbc:wm:sqlserver://10.60.72.87:1433;databaseName=SQL_DB"
```

Platform Manager での実行時の例

- インストールされた Integration Server に対し、インスタンス名 is-instance2、診断ポート 8083、JMX ポート 10058、プライマリポート 8081 の新しいインスタンスを作成するには

```
sagcc create instances integrationServer instance.name=is-instance2
diagnostic.port=8083 jmx.port=10058 primary.port=8081
db.alias=sql_server db.type=sqlserver
db.username=sa db.password=password@1234
db.url="jdbc:wm:sqlserver://10.60.72.87:1433;databaseName=SQL_DB"
```

- 現在のディレクトリにある instance-settings.properties ファイルのインスタンスデータを使用して、インストールされた Integration Server に対する新しいインスタンスを作成するには

```
sagcc create instances integrationServer -i instance-settings.properties
```

- 現在のディレクトリにある instance.settings.xml ファイルのインスタンスデータを使用して、インストールされた Integration Server に対する新しいインスタンスを作成するには

```
sagcc create instances integrationServer -i instance-settings.xml
```

Software AG Command Central の詳細については、Command Central のマニュアルを参照してください。

Command Central を使用した Integration Server Administrator へのアクセス

Command Central で、SSO (Single Sign-On : シングルサインオン) は、インストール後の設定なしに管理リンクを使用して webMethods 製品を管理するよう設計されています。拡張設定タスクを実行する場合、製品のプライマリ管理インタフェースへのアクセスが必要なことがあります。Command Central では、管理された各製品の [Instances Overview] ページに管理インタフェースへのリンクが用意されています。たとえば、Integration Server インスタンスの [Overview] ページで Integration Server リンクをクリックすると、Command Central は対応する Integration Server Administrator URL にブラウザをリダイレクトします。

Command Central Web ユーザインタフェースの Integration Server への管理リンクにアクセスするには、管理クレデンシャルが必要です。

重要:
特定のランドスケープを管理するには、1 つの Command Central インスタンスのみ使用してください。特定の Command Central インスタンスの Command Central Web ユーザインタフェースに別の Command Central インスタンスからアクセスすることはできません。

Software AG Common Platform ベースの製品のカスタム SSO および SAML 認証を生成し設定する方法の詳細については、Software AG のセキュリティインフラストラクチャのマニュアルを参照してください。

認証モードの変更

デフォルトで、信頼できる認証をサポートするランタイムコンポーネントの認証モードは [**Trusted**] に設定されます。

インスタンスの [Overview] タブで、[Authentication] フィールドの をクリックすることにより、[Authentication Mode] ダイアログボックスを使って認証モードを変更します。

Integration Server インスタンスの認証モードを指定した場合、その認証モードは主な製品インスタンスの階層化されたすべての製品インスタンスにも設定されます。ただし、Integration Server の OSGI プロファイルの認証モードを変更しても、OSGI プロファイルに属する Integration Server ランタイムコンポーネントの認証モードは変更されません。

基本認証を使用するには、ランタイムコンポーネントの認証モードを [**Fixed User**] に変更する必要があります。Command Central は固定ユーザに基本認証を使用して、Platform Manager と通信します。[**Fixed User**] 認証を使用すると、Platform Manager の認証クレデンシャルは固定されます。

Integration Server で Command Central の管理者ユーザパスワードを変更する

Command Central Web ユーザインタフェースで、Command Central によって管理されている Integration Server の管理者パスワードを変更します。Command Central で管理製品の管理者パスワードを変更した後、送信クレデンシャルは自動的に更新されます。

Command Central で製品の管理者ユーザパスワードを変更するには

1. Command Central の [Environments] パネルで、管理製品インスタンスを含む環境を選択します。

2. [Instances] テーブルで、integrationServer-default など、integrationServer-*instancename* の下にある Integration Server コンポーネントを選択します。

3. [Configuration] タブで、[**Users**] を選択します。

4. [Users] ページで、[**Administrator**] をクリックします。

5. [**Edit**] をクリックし、[**Administrator**] の新しいパスワードを指定します。

   新しいクレデンシャルを確認するには、 をクリックし、製品コンポーネントの監視 KPI をチェックします

SSL 接続のために Integration Server を設定する

Integration Server のプライマリポートが HTTPS ポートであり、そのクライアント認証が [**クライアント認証を必須にする**] に設定されている場合、Integration Server で以下の設定手順に従って、SSL 接続が正しいことを確認します。

以下の設定では、Integration Server 用の Platform Manager プラグインが、[**キーストアエイリアス**] と [**キーエイリアス**] に対応する認証を使用することを確認します。この認証は HTTPS ポートで設定され、SSL 接続時に Command Central から Integration Server に同じ認証を送信します。

SSL 接続のために Integration Server を設定するには

1. Integration Server HTTPS プライマリポートで設定されたトラストストアに認証を追加します。

2. 認証を Integration Server の管理者ユーザにマップします。

   認証をトラストストアに追加し、認証を Integration Server ユーザにマップする方法の詳細については、549 ページの「クライアントの認証」を参照してください。

Integration Server インスタンスの監視

IntegrationServer-*instanceName* のランタイム監視の状態

以下の表に、IntegrationServer-*instanceName* ランタイムコンポーネントが `sagcc get monitoring runtimestatus` および `sagcc get monitoring state` コマンドに応答して返すことができるランタイムの状態、およびランタイムの各状態の意味を示します。

ランタイムの状態	意味
ONLINE	Integration Server は稼働中であり、Integration Server プライマリポートで要求を受け入れ、処理しています。
PAUSED	Integration Server は休止モードになっています。Integration Server は診断ポートおよび休止ポートでのみ要求を受け入れ、処理しています。
STOPPED	Integration Server は停止しています。Integration Server は Integration Server プライマリポートで要求を受け入れず、処理していません。
UNKNOWN	Integration Server の状態は判別できません。
UNRESPONSIVE	Integration Server は稼働していますが、アクセスできません。 **注:** IS-*instanceName* は ONLINE を報告することがあります。この場合、Integration Server に問題があることを示します。

IntegrationServer-*instanceName* のランタイム監視の状態

`sagcc get monitoring runtimestate` および `sagcc get monitoring state` コマンドに応答して、IntegrationServer-*instanceName* ランタイムコンポーネントは以下のキーパフォーマンスインジケータ (KPI) に関する情報を提供します。

KPI	説明
実行中のサービス	この KPI を使用して、Integration Server が同時に実行しているサービスの数を監視し、その数がクリティカル値に近付いた場合に修正処置を実行できるようにします。実行中のサービスの数には、トリガーされたか、スケジュールされたか、または直接呼び出されたサービスが含まれます。 KPI は以下のマージナル値、クリティカル値、および最大値を使用します。 ■ マージナルは、同時実行中のサービスの最高水準点の 80% を示します。

KPI	説明
	■ クリティカルは、同時実行中のサービスの最高水準点の 95% を示します。 ■ 最大は、同時実行中のサービスの最高水準点の 100% を示します。これは Integration Server のスレッド領域に表示されます。
応答時間	この KPI を使用して、サービスの応答時間を監視し、応答時間がクリティカル値に近付いた場合に修正処置を実行できるようにします。 KPI は次のマージナル値、クリティカル値、および最大値を使用します。 ■ マージナルは、サービス応答時間の最高水準点の 80% を示します。 ■ クリティカルは、サービス応答時間の最高水準点の 95% を示します。 ■ 最大は、サービス応答時間の最高水準点の 100% を示します。
サービスエラー	この KPI を使用して、直前の 1 分間に発生したサービス例外の数を監視し、例外の現在の数がクリティカル値に近付いた場合に修正処置を実行できるようにします。 KPI は次のマージナル値、クリティカル値、および最大値を使用します。 ■ マージナルは直前の 1 分間で 5 個の例外を示します。 ■ クリティカルは直前の 1 分間で 20 個の例外を示します。 ■ 最大は直前の 1 分間で例外が 20 個を超えていることを示します。

Integration Server インスタンスの KPI の監視

Integration Server インスタンスの KPI を監視するには、以下の手順に従います。

Integration Server インスタンスの KPI を監視するには

1. [**Environments**] パネルで、監視する環境を選択します。
2. [**Instances**] タブをクリックします。
3. 表で、監視する Integration Server を選択します。
4. [**Overview**] タブをクリックします。

[**Dashboard**] の [**Monitoring**] セクションに、Integration Server インスタンスの KPI が表示されます。

Integration Server は以下の 3 つの KPI を返します。

名前	マージナル値	クリティカル値	最大値
[Average response time (in ms)]	最大値の 80%	最大値 の 95%	5000
[Service errors]	最大値の 70%	最大値 の 90%	5
[Running services]	最大値の 80%	最大値 の 95%	10

Integration Server の設定タイプ

IntegrationServer-*instanceName* がサポートする設定タイプ

IntegrationServer-*instanceName* (*instanceName* は Integration Server インスタンスの名前) ランタイムコンポーネントは、以下の表に示す設定タイプのインスタンスの作成をサポートしています。

設定タイプ	設定 ID	設定内容
管理 UI リンク	COMMON-ADMINUI	Integration Server への完全 URL。
クラスタリング	COMMON-CLUSTER	クラスタ設定を有効または無効にし、Terracotta Server Array の URL、セッションタイムアウト、および起動エラー時のアクションなどのクラスタ設定を指定する設定インスタンス。
データベースの機能エイリアス	COMMON-DBFUNCTION	Integration Server のデータベース機能エイリアスの設定インスタンス。
JDBC 接続プール	COMMON-JDBC	Integration Server のJDBC 接続プールの設定インスタンス。
JMS 設定	COMMON-JMS	Integration Server の JMS 設定の設定インスタンス。
JNDI 設定	COMMON-JNDI	Integration Server の JNDI 設定の設定インスタンス。
キーストアおよびトラストストアのエイリアス	COMMON-KEYSTORES	キーストアファイルまたはキーストア内の秘密鍵を識別するキーストアエイリアスの設定インスタンス。
LDAP ディレクトリ	COMMON-LDAP	Integration Server の LDAP ディレクトリ設定の設定インスタンス。

設定タイプ	設定 ID	設定内容
		各 LDAP 接続は異なる COMMON-LDAP 設定インスタンスを使用して宣言する必要があります。
Integration Server Core および Terracotta ライセンスファイル	COMMON-LICENSE	ライセンスファイル。IntegrationServer-*instanceName* は Integration Server Core および Terracotta ライセンスファイルの設定インスタンスをサポートしています。
Integration Server Core および Terracotta ライセンスファイル	COMMON-LICLOC	Integration Server のインストール先ファイルシステム内でライセンスファイルがある場所。IntegrationServer-*instanceName* は、Integration Server Core ライセンスファイルの場所および Terracotta ライセンスファイルの場所の設定インスタンスをサポートしています。 ライセンスファイルの場所は変更できません。
ユーザ管理	COMMON-LOCAL-USERS	ローカルユーザおよびそのパスワードを管理する設定インスタンス。COMMON-LOCAL-USERS-<*userId*> はさまざまなユーザの設定をサポートしています。各ユーザは異なる設定インスタンスを使用して宣言する必要があります。 デフォルトでは、COMMON-LOCAL-USERS-Administrator、COMMON-LOCAL-USERS-Default、COMMON-LOCAL-USERS-Developer、COMMON-LOCAL-USERS-Replicator という 4 つの事前定義済み COMMON-LOCAL-USERS 設定インスタンスがあります。 **注:** 作成時に、ユーザはすべて自動的に Everybody グループに追加されます。
Integration Server ロガーおよびサーバログ機能	COMMON-LOGGERS	Integration Server ロガーおよびサーバログ機能の設定インスタンス。 サーバロガーの場合、ロガーのレベルのデフォルト値は `Inherit` です。この値は、コンポーネントがその親と同じログレベルであることを示します。ロガーレベルの追加

設定タイプ	設定 ID	設定内容
		または削除はできず、レベル値の変更のみできます。
ポート	COMMON-PORTS	ポート。IntegrationServer-*instanceName* は HTTP、HTTPS、FTP、FTPS、および診断ポートの設定インスタンスをサポートしています。
プロキシサーバエイリアス	COMMON-PROXY	プロキシサーバエイリアスの設定インスタンス。 **注:** デフォルトプロキシエイリアス **proxy-bypass** を使用し、オプションとして、プロキシをバイパスして、選択した要求を直接そのターゲットにルーティングできます。プロキシバイパスアドレスリストは、すべてのプロキシサーバエイリアスに適用されます。**proxy-bypass** エイリアスの作成または削除はできません。
電子メールの設定	COMMON-SMTP	電子メール送信の設定。
サーバ設定パラメータ	IS-SYSPROPS	サーバ設定パラメータ。IntegrationServer-*instanceName* は、以下の server.cnf 設定ファイルで定義されたシステム設定パラメータの追加、更新、削除をサポートしています。 *Integration Server_directory* /instances/*instance_name*/config/server.cnf。*instance_name* は Integration Server インスタンスの名前です。 **注:** IS-SYSPROPS は Command Central テンプレート (YAML) によってのみサポートされます。
	COMMON-SYSPROPS	サーバ設定パラメータ。IntegrationServer-*instanceName* は、以下の server.cnf 設定ファイルで定義されたシステム設定パラメータの上書きをサポートしています。 *Integration Server_directory* /instances/*instance_name*/config/server.cnf。

設定タイプ	設定 ID	設定内容
		instance_name は Integration Server インスタンスの名前です。 **注:** Command Central ユーザインタフェースと Command Central テンプレートの両方で COMMON-SYSPROPS をサポートしています。
キーストアおよびトラストストアのエイリアス	COMMON-TRUSTSTORES	トラストストアファイルを識別するトラストストアエイリアスの設定インスタンス。
グローバル変数	COMMON-VARS	グローバル変数の設定インスタンス。各変数は異なる設定インスタンスを使用して宣言する必要があります。
webMethods メッセージング設定	COMMON-WMMESSAGING	Integration Server の webMethods メッセージング設定の設定インスタンス。IntegrationServer-*instanceName* は webMethods メッセージングプロバイダ (webMethods Broker および Universal Messaging) の設定インスタンスをサポートしています。 **注:** 接続状態が true として有効化された Command Central から Broker 接続エイリアスを作成した後、変更内容を有効にするには、Integration Server を再起動します。
Web サービスエンドポイントエイリアス - コンシューマ	IS-CONSUMER-ENDPOINTS	コンシューマ Web サービスエンドポイントエイリアス。
webMethods メッセージング設定	IS-DEFAULT-WMMESSAGING	デフォルトのメッセージング接続エイリアス。
スケジューラ	IS-SCHEDULED-TASKS	ユーザタスクをスケジュールする Integration Server スケジューラの設定インスタンス。
データベースの機能エイリアス	IS-FILEPERMISSION	WmPublic パッケージの pub.file サービスのファイルアクセスコントロール設定。IS-FILEPERMISSION 設定インスタンスには、IS-FILEPERMISSION_READ、IS-FILEPERMISSION_WRITE、および IS-FILEPERMISSION_DELETE が含まれます。

設定タイプ	設定 ID	設定内容
		新しい設定インスタンスは作成できません。また、既存の設定インスタンスは削除できません。ファイルアクセスコントロール設定の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。
ポート	IS-PRIMARYPORT	Integration Server のプライマリポート ID。
Web サービスエンドポイントエイリアス - プロバイダ	IS-PROVIDER-ENDPOINTS	プロバイダ Web サービスエンドポイントエイリアス。
休止設定オプション	IS-QUIESCE-OPTIONS	Integration Server の休止オプションの設定インスタンス。
ポート	IS-QUIESCEPORT	Integration Server の休止ポート ID。
リソース設定	IS-RESOURCES	Integration Server のリソース設定の設定インスタンス。 IntegrationServer-*instanceName* は送信 HTTP、ステートフルセッションの制限、サーバスレッドプール、ドキュメントストア、および XA 回復ストアの設定の設定インスタンスをサポートしています。
JAAS 領域設定	COMMON-JAAS-REALMS	JAAS 領域で作成、読み取り、更新、および削除の各操作を実行する設定インスタンス。 Integration Server では、次の事前定義済み JAAS-REALMS 設定インスタンスをサポートしています。 IS_KERBEROS_INBOUND IS_KERBEROS_OUTBOUND IS_Transport PlatformManagement WSS_Message_IS WSS_Transport_IS WSS_Transport_ProxyService カスタムログインモジュールに領域を追加する場合、必ず次の形式を使用してください。

設定タイプ	設定 ID	設定内容

		customRealm{
		${loginModuleName} Flag;
		};
		
		注:
		入力値が正しい形式でない場合、次のエラーが発生し、Integration Server は起動に失敗します。
		javax.security.auth.login.LoginException: No LoginModules configured for ${realm_Name}”

グループ管理	IS-GROUPS	グループ管理のローカルグループおよびメンバーユーザを管理する設定インスタンス。LDAP および CDS グループはこの設定インスタンスを使用して表示されますが、作成、編集または削除することはできません。 IS-GROUPS-<GroupType>_<groupName> はさまざまなグループの設定をサポートしています。 各グループは異なる設定インスタンスを使用して宣言する必要があります。<GroupType> は、ローカルグループに対応する「local」 / LDAP グループに対応する「LDAP」 / CDS グループに対応する「system」です。 Integration Server は、次の事前定義済み IS-GROUPS 設定インスタンスをサポートしています。 IS-GROUPS-local_Administrators IS-GROUPS-local_Anonymous IS-GROUPS-local_Developers IS-GROUPS-local_Everybody IS-GROUPS-local_Replicators.
ACLs Management	IS-ACLS	ACL およびメンバーグループを管理する設定インスタンス。 IS-ACLS-<aclName> はさまざまな ACL の設定をサポートしています。各 ACL は異なる設定インスタンスを使用して宣言する必要があります。

設定タイプ	設定 ID	設定内容
		Integration Server は、次の事前定義済み IS-ACLs 設定インスタンスをサポートしています。 IS-ACLS-Administrators IS-ACLS-Anonymous IS-ACLS-Default IS-ACLS-Developers IS-ACLS-Internal IS-ACLS-Replicators
SFTP Server	IS-SFTP-SERVERALIAS	Integration Server の SFTP サーバ設定の設定インスタンス。
SFTP User	IS-SFTP-USERALIAS	Integration Server の SFTP ユーザ設定の設定インスタンス。

Integration Server の設定タイプの使用

Command Central 上で Integration Server 設定タイプの項目の追加、編集、または削除を行うには、以下の手順に従います。

注:
以下の手順に従う前に、Integration Server が稼働していることを確認します。

Integration Server 設定タイプの項目を追加、編集、または削除するには

1. [Environment] パネルから Integration Server 環境を選択し、[**Instances**] タブでインスタンスをクリックします。
2. [**Configuration**] タブをクリックします。
3. ドロップダウンリストから設定タイプを選択します。

   Command Central は使用可能またはデフォルトの値を表示します (選択した Integration Server 設定タイプの値がある場合)。
4. Integration Server 設定タイプの項目を追加するには、 をクリックします。表示されたフィールドに必要な値を入力し、[**Save**] をクリックします。
5. 設定タイプの項目を編集するには、更新する項目をクリックし、[**Edit**] をクリックします。必要な変更を行い、以下のいずれかをクリックします。

- [**Test**]: 設定タイプ項目をテストする場合
- [**Save**]: 変更内容を保存する場合
- [**Cancel**]: 設定タイプ項目の編集をキャンセルする場合

6. 設定インスタンスの項目を削除するには、 をクリックします。

Integration Server のライフサイクルアクション

以下の表に、`sagcc exec lifecycle` コマンドで Integration Server がサポートするアクション、およびアクションの実行時に Integration Server に対して実行される操作を示します。これらのアクションは Command Central Web ユーザインタフェースを使用して実行することもできます。.

アクション	説明
`start`	Integration Server インスタンスを開始します。
`stop`	Integration Server インスタンスを停止します。
`restart`	Integration Server インスタンスを再開します。
`pause`	アクティブな Integration Server を休止モードに設定します。作業中のサービスが完了し、パッケージがすぐに無効にならないよう、Integration Server は休止モードに入る前に 1 分間待機します。Integration Server の実行時の状態は PAUSED に設定されます。Integration Server が一時停止された場合、診断ポートと休止ポート以外のすべてのポートは無効になります。休止モードでは、既に進行中であったすべての要求は完了することを許可されますが、サーバへの新しい受信要求はブロックされます。JDBC プールへの接続や LDAP またはセントラルユーザディレクトリによる接続などの送信接続試行は、開いたままになります。
`resume`	休止モードの Integration Server をアクティブモードに切り替え、通常動作を再開します。無効化または一時停止されたすべてのアセットやアクティビティは復元または再開されます。Integration Server の実行時の状態は ONLINE に戻ります。

Integration Server でサポートしているコマンド

Integration Server は以下の Platform Manager コマンドをサポートしています。

- `sagcc create configuration data`
- `sagcc delete configuration data`
- `sagcc get configuration data`
- `sagcc update configuration data`

- sagcc get configuration instances
- sagcc list configuration instances
- sagcc get configuration types
- sagcc list configuration types
- sagcc exec configuration validation create
- sagcc exec configuration validation delete
- sagcc exec configuration validation update
- sagcc create instances
- sagcc delete instances
- sagcc list instances supported products
- sagcc update instances
- sagcc get inventory components
- sagcc list inventory components
- sagcc exec lifecycle
- sagcc get monitoring
- sagcc list administration

上記のコマンドの詳細については、『Software AG Command Central Help Guide』を参照してください。

UNIXシェルスクリプトを使用した管理製品の接続クレデンシャルの変更

次のサンプル UNIX シェルスクリプトを使用して、Command Central が管理する製品コンポーネントの基本認証クレデンシャルを設定できます。

```
NODE_ALIAS=local
USERNAME=Administrator
PASSWORD=secret
RCID=integrationServer-default

sagcc get configuration data $NODE_ALIAS $RCID COMMON-LOCAL-USERS-Administrator

-o administrator.xml
sed "s,/>,><Password>${PASSWORD}</Password></User>,g" administrator.xml >

 administrator_new.xml
sagcc update configuration data $NODE_ALIAS $RCID COMMON-LOCAL-USERS

-Administrator -o administrator_new.xml
```

```
# verify connection
sagcc get monitoring runtimestatus $NODE_ALIAS $RCID -e ONLINE
```

Command Central コマンドラインツールのアップグレード

Command Central REST API またはコマンドラインツールコマンドを使用する、既存のすべての Integration Server 自動化スクリプトおよびクエリーで、すべての Integration Server インスタンスの Integration Server インスタンス ID を変更します。

```
runtimeComponent=integrationServer-ENGINE
```

このインスタンス ID を以下のように変更します。

```
runtimeComponent=integrationServer-instanceName
```

instanceName は Integration Server インスタンスの名前です。例: `integrationServer-default`

A Integration Server の展開チェックリスト

- はじめに 1086
- ステージ 1: インストール 1086
- ステージ 2: 基本設定 1087
- ステージ 3: ユーザ、グループおよび ACL の設定 1088
- ステージ 4: パッケージのパブリッシュ 1089
- ステージ 5: 実行時クラスのインストール 1090
- ステージ 6: サーバと通信するためのクライアントの準備 1090
- ステージ 7: セキュリティの設定 1091
- ステージ 8: 起動とテスト 1093
- ステージ 9: ソースのアーカイブ 1093

はじめに

この付録には、webMethods Integration Server の設定に役立つチェックリストが含まれています。ここでは Integration Server の導入手順について説明します。この手順はいくつかのステージから成ります。各ステージを完全に終了してから次のステージに進んでください。

ステージ 1: インストール

以下の手順に従って Integration Server のインストール、実行およびテストを行います。

手順	アクション	☒
1.	**Integration Server をインストールします。** 手順については、『*Installing Software AG Products*』を参照してください。 **注:** Integration Server は、Windows アプリケーションまたは Windows サービスとしてインストールすることができます。インストール後、必要に応じて Windows アプリケーションから Windows サービスに切り替えたり、その逆を行うことが可能です。手順については、『45 ページの「Windows アプリケーションまたは Windows サービスとしての Integration Server の実行」』を参照してください。	
2.	**デフォルトパスワードを変更します。** Integration Server Administrator を使用して、次のユーザアカウントに新しいパスワードを割り当てます。 ■ Administrator ユーザアカウント ■ Developer ユーザアカウント ■ Central ユーザアカウント ■ Replicator ユーザアカウント パスワードの変更方法については、106 ページの「Administrator ユーザの追加」を参照してください。 Integration Server Administrator を使用して、送信パスワードを格納する前の暗号化の際に使用する Integration Server の新しいマスターパスワードを割り当てます。マスターパスワードの変更方法については、574 ページの「マスターパスワードの変更」を参照してください。	
3.	**送信パスワードおよびマスターパスワードの使用方法を決定します。** 初めて Integration Server を起動して設定を行う前に、Integration Server で送信パスワードとマスターパスワードをどのように扱うか (格納場所、暗号化方法、変更頻度) を決定します。Integration Server を設定した後でこれらの設定を変更すると、マスターパスワードと送信パスワードの同期が	

手順	アクション	☒
	外れることがあります。詳細については、483 ページの「Integration Server との通信のセキュリティの設定」を参照してください。	
4.	**JDK または JRE を使用して Java の場所を指定するかどうかを決定します。** Designer や Developer を使用して Integration Server の Java サービスを開発およびコンパイルする場合は、JDK の場所を指定します。Java サービスのコンパイルに Integration Server のこのインストールを使用しない場合は、JRE または JDK の場所を指定できます。Java の場所の指定方法の詳細については、52 ページの「Integration Server への Java システムプロパティの受け渡し 」を参照してください。	

ステージ 2: 基本設定

Integration Server Administrator を使用して、サーバによる送信要求の送信、受信要求の受信、セッションの失効およびエラーメッセージの発行に関する設定を行います。

手順	アクション	☒
1.	**ポートを設定します。** [ポート] ページで、要求を受信待機するサーバ上のポートを指定します。 複数のポートで HTTP 要求と HTTPS 要求の両方またはいずれか一方を受信する場合は、実稼働用にサーバの準備が整うまで Integration Server Administrator とのやり取りに使用するポート以外はすべて無効にすることができます。 ポートの設定および無効化の方法については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。	
2.	**プロキシサーバを指定します。** このサーバがプロキシサーバを介して送信要求を発行する場合は、[**プロキシサーバ**] ページでそのサーバを指定します。 必要であれば、プロキシサーバを迂回できる URL を指定します。 プロキシサーバとバイパスリストの指定方法については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。	
3.	**セッションのタイムアウトを設定します。** [**リソース**] ページで、サーバが使用するタイムアウト値を設定します。 手順については、『128 ページの「サーバセッションの管理」』を参照してください。	
4.	**エラーメッセージの受信者と SMTP サーバを指定します。**	

手順	アクション	☒
	[**ログ設定**] ページで、例外 (重大なサーバエラーやバインディングの失敗) 発生時にエラーメッセージを送信する宛先の電子メールアドレス、およびそのために使用する SMTP サーバの名前を指定します。 手順については、145 ページの「Integration Server によるログ、状態、その他の情報の保存場所の設定」を参照してください。	
5.	**ログを設定します。** 手順については、『*webMethods Audit Logging Guide*』および235 ページの「サーバログの設定」を参照してください。	

ステージ 3: ユーザ、グループおよび ACL の設定

Integration Server Administrator で、ユーザアカウント、グループおよび ACL を指定して、このサーバで実行するサービスへの適正なアクセスレベルを設定します。

手順	アクション	☒
1.	**サービスのセキュリティ要件を指定します。** 管理者および開発者以外のユーザによるサービスへのアクセスは暗黙的にブロックされます。アクセスレベルを指定し (1 つのユーザグループ、すべての認証ユーザまたは未認証ユーザにもアクセスを許可するかどうか)、適正な ACL をサービスに適用します。	
2.	**ユーザ ID とグループを作成するか、または外部ディレクトリを設定します。** セキュアサービスを保持している場合は、これらのサービスへのアクセスを許可するユーザとクライアントアプリケーションの両方またはいずれか一方を識別し、これらの許可メンバーを含むグループを作成します。 使用サイトで外部ディレクトリ (LDAP またはセントラルユーザ管理) を使用する場合は、サーバから外部ディレクトリのユーザ情報およびグループ情報にアクセスするように設定できます。 ユーザ ID の作成方法については、104 ページの「ユーザアカウントの追加」を参照してください。グループの作成方法については、118 ページの「グループの追加」を参照してください。外部ディレクトリの使用方法については、675 ページの「セントラルユーザディレクトリまたは LDAP の設定」を参照してください。	
3.	**ACL を作成します。** サービスのセキュリティ要件を満たすために必要な ACL を作成し、作成済みのグループをその ACL に割り当てます。手順については、『539 ページの「ACL の作成」』を参照してください。	

手順	アクション	☒
4.	**補助管理者を指定します。** 主管理者がその役割を実行できない場合に、補助管理者として作業できるユーザを 1 人か 2 人選任します。[**ユーザとグループ**] ページでこのユーザを Administrators グループに追加します。 ユーザに Administrator 特権を付与する方法については、106 ページの「Administrator ユーザの追加」を参照してください。	

ステージ 4: パッケージのパブリッシュ

サーバ上で実行するパッケージのインストールと設定を行います。

手順	アクション	☒
1.	**サーバにサービスをインストールします。** 次のいずれかの方法で製品サーバにサービスをパブリッシュします。 ■ *方法 1:*[**パッケージ**] > [パブリッシュ] ページで、開発サーバから製品サーバにパッケージを複製します。 手順については、『716 ページの「サーバ間でのパッケージのコピー」』を参照してください。 ■ 方法 2:パブリッシャーサーバの Integration Server Administrator で、パブリッシュする各パッケージを含めた圧縮ファイルを作成します。 1. 次に、ターゲットサーバ上にある次のディレクトリに圧縮ファイルをコピーします。 *Integration Server_directory*\instances*instance_name*\replicate\inbound 2. [**パッケージ**] > [管理] ページで各パッケージをインストールします。	
2.	**サーバ上でサービスを設定します。** 各サービスが有効であることを確認します。次に、サービスごとに下記の操作パラメータを設定します。 ■ ACL の割り当て 手順については、『542 ページの「フォルダ、サービス、その他のエレメントへの ACL の割り当て」』を参照してください。 ■ キャッシングパラメータ 手順については、『*webMethods Service Development Help*』を参照してください。	

手順	アクション	☒
	■ 出力テンプレートの割り当て 手順については、『*webMethods Service Development Help*』および『*Dynamic Server Pages and Output Templates Developer's Guide*』を参照してください。	

ステージ 5: 実行時クラスのインストール

Java や Integration Server で提供される実行時クラス以外に、CORBA クラスまたは MQ Series クラスなどの別のクラスをサービスで使用する場合は、そのクラスをサーバにインストールします。

手順	アクション	☒
1.	**実行時クラスをインストールします。** ベンダーから zip または jar ファイルを取得し、Integration Server からアクセスできるデバイスまたはディレクトリに zip または jar ファイルをコピーします。	
2.	**クラスパスを更新します。** custom_wrapper.conf ファイル内の `wrapper.java.additional` プロパティの classpath ステートメントを更新して、それが実行時クラスのインストール先のディレクトリを指し示すようにします。 Microservices Runtime の場合、setenv.bat/sh で Java システムプロパティを更新します。 クラスパスの更新方法の詳細については、968 ページの「クラスをサーバクラスパスに追加する」を参照してください。	

ステージ 6: サーバと通信するためのクライアントの準備

Integration Server クライアントとして使用するアプリケーション (Java、C/C++ プログラムなど) がある場合は、Integration Server と通信するためにクライアントの準備を行う必要があります。

手順	アクション	☒
1.	Integration Server の client.jar ファイルには、Integration Server と通信するためにクライアントに必要なクラスが含まれています。Integration Server または Designer がインストールされているコンピュータにクライアントが存在する場合は、wm-isclient.jar ファイルを含むようにコンピュータのクラスパスを設定します。wm-isclient.jar ファイルは common\lib ディレクトリに	

手順	アクション	☒

配置されているので、クラスパスを %CLASSPATH%; *Software AG_directory* \common\lib\wm-isclient.jar と設定します。

2. Integration Server または Designer をホストしないコンピュータにもクライアントが存在する場合は、各コンピュータに対して以下の作業を行います。

   1. *Software AG_directory* \common\lib ディレクトリに移動し、wm-isclient.jar ファイルをクライアントコンピュータの任意のディレクトリにコピーします。

   2. SSL を使用して Integration Server と通信する場合は、*Software AG_directory* \common\lib\ext ディレクトリに移動し、enttoolkit.jar ファイルをクライアントコンピュータの任意のディレクトリにコピーします。

   3. クライアントコンピュータで、wm-isclient.jar ファイルと enttoolkit.jar ファイル (該当する場合) を含むようにクラスパスを設定します。たとえば、wm-isclient.jar ファイルと enttoolkit.jar ファイルを c:\myapp ディレクトリに配置した場合は、クラスパスを %CLASSPATH%;c:\myapp\client.jar;c:\myapp\enttoolkit.jar と設定します。

ステージ 7: セキュリティの設定

以下の手順に従って、使用するセキュリティ対策が適切であることを確認します。

手順	アクション	☒

1. **パスワードをチェックします。**

   Administrator アカウントおよび Replicator アカウントのパスワードと、送信パスワードの暗号化のマスターパスワードが、webMethods Integration Server によって割り当てられたデフォルト値から変更されていることを確認します。

2. **index.html ファイルを編集して Integration Server Administrator へのアクセスを防止します。**

   ユーザが誤って Integration Server Administrator にアクセスするのを防ぐ場合は、次のファイルを編集します。

   Integration Server_directory \instances*instance_name*\packages\Default\pub\index.html

   <frame src="/WmRoot/index.dsp"\> タグ内の SRC を、作成済みの無難なページ (代替リンクが設置された、エラーメッセージを表示するページなど) に変更します。

手順	アクション	☒

Integration Server のすべての DSP ファイルがユーザに表示されないようにするには、watt.server.displayDirectories サーバ設定パラメータを使用します。

index.html ファイルを編集すると、通常の方法 (単純に Integration Server の受信待機ポートに接続する) では Integration Server Administrator を呼び出せなくなるので注意してください。この場合には、次のような Integration Server Administrator の完全 URL を入力する必要があります。

```
http://Server:Port/WmRoot/index.dsp
```

ここで、

Server は Integration Server の名前であり、

Port は HTTP 要求の受信待機ポートです。

3. **ユーザアカウントをチェックします。**

   すべてのユーザアカウントに、必要とされるパスワードが存在することを確認します。

4. **ACL の割り当てをチェックします。**

   すべてのセキュアサービスに適正な ACL が割り当てられていることを確認します。

5. **プロキシサーバの設定をチェックします。**

   プロキシサーバの設定およびバイパスリストが正しいことを確認します。

6. **アクセスを制限します。**

   必要に応じて、受信要求を制限するために許可/拒否リストを設定します。

7. **デジタル認証をインストールおよび設定します。**

8. **HTTP ルーティングシステムを設定します。**

   サーバがルーティングシステム、負荷分散システムまたは URL フィルタリングシステムの後方に配置されている場合は、そのシステムの管理者に問い合わせて、Integration Server への受信要求がそのシステムを通過できることを確認します。

9. **特定の管理者ユーザの maskSessionID.properties ファイルへの書き込み権限を設定します。**

   特定の管理者ユーザにのみ、maskSessionID.properties ファイルへの書き込み権限を付与します。残りの管理者ユーザには、このファイルへの読み取り権限のみ設定する必要があります。

10. **オペレーティングシステムのセキュリティを確認します。**

手順	アクション	☒
	Integration Server のセキュリティは、オペレーティングシステムのセキュリティに依存しています。したがって、オペレーティングシステムが正しく設定されているか、すべてのセキュリティパッチが適用されているか、および telnet やメールなどの不要なネットワークサービスが削除されているかを確認する必要があります。	

ステージ 8: 起動とテスト

サーバを起動してサービスをテストし、必要な動作をするか確認します。

手順	アクション	☒
1.	**ポートが有効になっていることを確認します。** セットアップ中のサーバへのアクセスを禁止するためにポートを無効にしている場合は、ここで [**ポート**] ページを使用して有効にします。 **ヒント:** ポートを有効にしたら、ネットワークの接続を確認し、接続が有効になっていることを確認します。	
2.	**サーバを再起動します。** Integration Server Administrator を使用してサーバを再起動し、設定がすべて有効であることを確認します。 手順については、『56 ページの「Integration Server の再起動 」』を参照してください。	
3.	**サービスをテストします。** テストを実行して、ユーザ/クライアントアプリケーションがサーバに正常にアクセスできることを確認します。 **注:** このテスト中に、現在のライセンスが、このサーバで発生する可能性のある同時要求に対応しているかどうかを確認することもできます。ライセンスで認可されているセッションの数を増やす必要がある場合は、Software AG までお問い合わせください。	
4.	**サーバを実際に稼動させます。**	

ステージ 9: ソースのアーカイブ

サーバ上のパッケージのマスターコピーおよびパッケージの作成に使用されたソースファイルをアーカイブします。

手順	アクション	☒
1.	server\packages ディレクトリの内容をバックアップまたはアーカイブするために、別のデバイスにコピーします。	
2.	このサーバに展開されたサービスの作成に使用された全ソースファイルのコピーをアーカイブします。	

B サーバ設定パラメータ

- はじめに 1097
- watt.adapters. 1097
- watt.adminapi. 1097
- watt.art. 1098
- watt.broker. 1099
- watt.brokerCoder. 1099
- watt.cachedirective. 1100
- watt.cds. 1100
- watt.client. 1100
- watt.config. 1102
- watt.core. 1102
- watt.debug. 1109
- watt.debug2. 1111
- watt.frag. 1112
- watt.infradc. 1112
- watt.net. 1112
- watt.security. 1129
- watt.server. 1136
- watt.serverlog. 1276

- watt.ssh. 1277
- watt.ssl. 1278
- watt.tx. 1279
- watt.um 1280
- watt.wm.tnextdc. 1282
- watt.wmcloud. 1282

はじめに

この付録では、 *Integration Server_directory* \instances*instance_name*\config ディレクトリ内のサーバ設定ファイル (server.cnf) に指定するパラメータについて説明します。このファイルを更新する場合、通常は Integration Server Administrator の [**設定**] > [拡張設定] ページを使用しますが、テキストエディタでファイルを直接編集することが必要になる場合もあります。ファイルを直接編集する場合は、最初に Integration Server をシャットダウンしてからファイルを更新します。変更を加えた後に、Integration Server を再起動します。

注:
[**設定**] > [拡張設定] ページを使用してサーバ設定ファイル (server.cnf) を更新する場合は、特に指定のない限り、サーバの再起動は必要ありません。設定の変更を反映するために再起動が必要となる変更が行われた場合は、Integration Server Administrator により「変更を反映するには再起動する必要があります。」というメッセージが表示されます。

Integration Server は、数々のパラメータに対してデフォルト値を使用します。パラメータにデフォルトがある場合は、パラメータの説明にデフォルトを記載しています。デフォルトがあるパラメータの多くは、Integration Server によって Integration Server Administrator を管理するときに設定されます。

watt.adapters.

watt.adapters.nlineHelp
オンラインヘルプコンテンツがあるアダプタ名のカンマ区切りリストを指定します。オンラインヘルプには、Integration Server Administrator のヘルプリンクからアクセスできるようにする必要があります。このパラメータのデフォルト値はありません。

watt.adminapi.

watt.adminapi.group.readOnly
メンバーが Integration Server Administrator API に読み取り専用アクセスを行うユーザグループの名前を指定します。メンバーは GET 要求のみを使用して Administrator API にアクセスできます。ユーザグループ名は大文字と小文字を区別します。このパラメータのデフォルト値はありません。

注:
WmAdmin パッケージを再ロードして、このパラメータへの変更事項を反映させます。または、Integration Server を再起動します。

watt.adminapi.log.clientErrors
Administrator API の REST リソースの実行中に発生するクライアントエラーをエラーログに記録するかどうかを指定します。クライアントエラー (400 番代の HTTP ステータスコードとなるエラー) をエラーログに書き込むには、このパラメータを「true」に設定します。クライアントエラーをエラーログに書き込まない場合、このパラメータを「false」に設定します。デフォルトは「false」です。

watt.adminapi.returnExceptions
Integration Server Administrator で例外が発生したときに送信される HTTP 応答の応答本文にスタックトレースを含めるかどうかを指定します。例外メッセージに加えてスタックトレー

スを応答本文に含めるには、このパラメータを「true」に設定します。スタックトレースを省略し、例外メッセージのみ返すには、このパラメータを「false」に設定します。デフォルトは「false」です。

watt.art.

注:
アダプタに関連する他のサーバ設定パラメータについては、アダプタのマニュアルを参照してください。

watt.art.analysis
ログを有効にしてアダプタリスナーとそのリンク通知を分析するかどうかを指定します。「true」に設定すると、wm.art.dev.notification:analyzeListenerNotifications サービスと wm.art.dev.listener:analyzeListenerNodes サービスはデータをサーバログファイルに記録します。wm.art.dev.listener:updateRegisteredNotifications サービスは、見つからないリスナー通知のリスナーを更新します。「false」に設定すると、分析が行われず、情報はログに記録されません。デフォルトは「false」です。

watt.art.connection.nodeVersion
アダプタ接続でパスワードを passman ストアに、パスワードハンドルを接続ノードに保存するかどうかを指定します。watt.art.connection.nodeVersion パラメータが「1」に設定されている場合、パスワードはアダプタ接続内に組み込まれます。このパラメータが「2」に設定されている場合、パスワードハンドルがアダプタ接続内に保存されます。デフォルトは 2 です。Software AG ではデフォルト値の使用をお勧めします。このパラメータに新しい値を設定するごとに、Integration Server を再起動するか WmART パッケージを再ロードする必要があります。

watt.art.connection.nodeVersion パラメータの値が「2」に設定されている場合、Deployer を使用したランタイムベースの展開中に、パスワードフィールドの変数の置換を実行して、パスワードをターゲットシステムに展開する必要があります。

watt.art.notification.publishToJMS.useCSQ
JMS プロバイダへのパブリッシュ時に Adapter Runtime の非同期通知でクライアントサイドキュー (CSQ) を使用できるようにするかどうかを指定します。「true」に設定した場合、JMS プロバイダを使用できないとき、JMS プロバイダに送信される非同期通知は CSQ にパブリッシュできます。デフォルトは「false」です。

watt.art.page.size
アダプタの [接続] ページ、[リスナー] ページおよび [通知] ページに表示される最大項目数を指定します。デフォルトは 10 です。ページネーションの制御の詳細については、アダプタのマニュアルを参照してください。

watt.art.synchronousNotification.selectExecuteUser
[リスナー通知] ページに [**実行ユーザ**] 列を含める WmArt ベースのアダプタを指定します。この列を所定の位置に配置すると、ユーザを通知に割り当てることができます。これにより、リスナー通知によってサービスが呼び出されると、指定したユーザとして実行されるようになります。1 つまたは複数のアダプタを指定できます。複数のアダプタを指定する場合は、次の例のように名前をセミコロン (;) で区切ります。

watt.art.synchronousNotification.selectExecuteUser=WmMQAdapter;WmSAP

watt.art.service.pipeline.hidden

アダプタサービスパイプラインを Integration Server ログファイルに記録するかどうかを指定します。watt.art.service.pipeline.hidden パラメータを `true` に設定すると、サービスパイプラインは Integration Server ログファイルに記録されません。このパラメータを `false` に設定すると、サービスパイプラインは Integration Server ログファイルに記録されます。デフォルトは「`false`」です。

watt.broker.

watt.broker.sync.enableBrokerSync
Integration Server が Broker クライアントと同期するかどうかを示します。このプロパティを「`true`」 (デフォルト) に設定した場合、Integration Server は Broker クライアントと同期します。値が「`false`」の場合、Integration Server は Broker クライアントと同期せず、同期がオフであることを示す警告メッセージをログに記録します。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.broker.sync.forceDispatcherInit
Integration Server が Broker 設定を初期化するかどうかを示します。この値を「`true`」に設定した場合、Integration Server は Broker 設定を再初期化します。Integration Server は、設定ファイルを利用できる場合は設定ファイルから再初期化します。設定ファイルを利用できない場合、Integration Server はデフォルト値で Broker 設定を初期化します。デフォルトは「`false`」です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.brokerCoder.

watt.brokerCoder.verbose
Integration Server が BrokerCoder の詳細ログを有効にするかどうかを示します。値を「`true`」に設定した場合、Integration Server で BrokerCoder の詳細ログが有効になります。デフォルトは「`false`」です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.cachedirective.

watt.cachedirective.exclude.packages
ダイナミックサーバページをブラウザでキャッシュするパッケージのカンマ区切りのリストを指定します。パッケージを正規表現として指定できます。正規表現に使用できるワイルドカード文字は、アスタリスク (*) のみです。デフォルトで、このパラメータの値は空です。これは、キャッシュされる Integration Server Administrator 内のダイナミックサーバページがないことを意味します。

注:
Software AG では、カスタムパッケージのみに関連するダイナミックサーバページをキャッシュするときは、watt.cachedirective.exclude.packages を使用することをお勧めします。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.cds.

watt.cds.skip.role.types
サーバの LDAP クエリー役割またはデータベースクエリー役割が Integration Server によって評価されるかどうかを制御します。Common Users が有効な Integration Server で、ACL 管理に LDAP クエリー役割またはデータベースクエリー役割が使用されていない場合は、これらの役割の評価が Integration Server のパフォーマンスに影響する可能性があります。この場合は、パフォーマンスを高めるために、クエリー役割の評価機能を無効化できます。詳細については、680 ページの「My webMethods Server クエリー役割に関する考慮事項」を参照してください。

LDAP クエリー役割の評価を無効にするには、`watt.cds.skip.role.types=wm_xt_ldapqueryprovider` を設定します。

データベースクエリー役割の評価を無効にするには、`watt.cds.skip.role.types=wm_xt_dbrole` を設定します。

両方のクエリー役割を無効にするには、上記の両方の設定を使用し、拡張設定を 2 回入力します。たとえば、[**拡張設定の編集**] ページで、次の情報を入力します。

```
watt.cds.skip.role.types=wm_xt_ldapqueryprovider
watt.cds.skip.role.types=wm_xt_dbrole
```

評価を再有効化するには、一方または両方の拡張設定を削除します。このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.client.

watt.client.data.blockSize
XML を解析するときに、Integration Server によって使用されるバッファサイズを定義します。以前、この値は 8192 バイトに設定されており、変更できませんでした。バッファサイズを大きくすると、XML の解析中のパフォーマンスが向上することがあります (XMLCoder および IDataXMLCoder)。内部テストによると、値が 65,536 以上になると、パフォーマンスはほとんど向上していないことに注意してください。このパラメータに設定する値は 2 の累

乗にする必要があります。最小値は 8192 です。値は、2 の累乗でない場合、Integration Server では無視され、デフォルト値が使用されます。デフォルトは 32768 です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.client.data.classLoader.legacy
パーシストされた XML をデコードするときに、Integration Server がクラスをキャッシュするかどうかを指定します。パーシストされた XML データのデコード時に Integration Server でクラスをキャッシュするには、「true」に設定します。パーシストされた XML データのデコード時にクラスをキャッシュすると、大きく複雑な XML データのデコード時にパフォーマンスが向上することがあります。キャッシュは単一のデコード操作時にのみ使用されるため、パフォーマンスが大幅に向上するのは、非常に大きく複雑な XML でのみです。Integration Server でデコード時にクラスをキャッシュしない場合、「false」に設定します。デフォルトは「true」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.client.data.maxToStringLength
IData.toString() によって作成される String の最大長を指定します。String の最大長に達すると、Integration Server は値をトリミングし、最後に省略記号 … を追加します。`watt.client.data.maxToStringLength` パラメータを小さい値に設定すると、非常に大きい IData オブジェクトで IData.toString() を使用するときに発生するパフォーマンスの問題が軽減することがあります。最小値は、0 より大きくする必要があります。最大値は 65536 です。`watt.client.data.maxToStringLength` の値が大きいほど、IData.toString() のパフォーマンスは低下することに注意してください。デフォルトは 1024 です。

注:
IData.toString() 応答の String の最大長の制限は、IData.toString() を使用して 2 つの String が同等であるかどうかを判断するソリューションに影響します。2 つの IData オブジェクトが同等であるかどうかを判断するには、equals() メソッドを使用します。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.client.hashcode.legacy
Integration Server で IData オブジェクトが等しいかどうかを判断するときに、Integration Server がレガシー実装 (Integration Server バージョン 10.11 より前) を使用するか現在の実装 (Integration Server 10.11 以降) を使用するかを指定します。現在の実装を使用すると、大きく複雑なデータを処理するフローサービスのパフォーマンスが向上することがあります。現在の実装を使用するには、「false」を設定します。レガシー実装を使用するには、「true」を設定します。デフォルトは「false」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.client.ftp.session.logoutOnServiceCompletion
トップレベルサービスの実行が完了したときに、pub.client.ftp:login サービスによって作成された FTP(S) セッションを自動的に閉じるかどうかを指定します。値を「`true`」に設定する

と、トップレベルサービスの実行が完了したときに、pub.client.ftp:login サービスによって作成された FTP(S) セッションは自動的に閉じます。値を「false」に設定した場合、pub.client.ftp:logout サービスを呼び出して、FTP(S) をセッションを閉じる必要があります。デフォルト値は「false」です。

watt.client.sftp.session.logoutOnServiceCompletion
トップレベルサービスの実行が完了したときに、pub.client.sftp:login サービスによって作成された SFTP セッションを自動的に閉じるかどうかを指定します。値を「true」に設定すると、トップレベルサービスの実行が完了したときに、pub.client.sftp:login サービスによって作成された SFTP セッションは自動的に閉じます。値を「false」に設定した場合、pub.client.sftp:logoutサービスを呼び出して、SFTP をセッションを閉じる必要があります。デフォルト値は「false」です。

watt.config.

watt.config.systemProperties
名前が watt 以外で始まる追加のシステムパラメータのリストを指定します。追加のシステムプロパティはそれぞれカンマで区切ります。デフォルトでは、mail.imap.partialfetch プロパティは追加のシステムプロパティとして含まれており、デフォルト値の「true」が設定されています。

watt.core.

watt.core.brokerTypeCoder.verbose
BrokerTypeCoder の詳細ログを有効または無効にします。「true」に設定すると、Integration Server は BrokerTypeCoder の詳細ログを有効にします。デフォルトは「false」です。

> **注:**
> webMethods Broker が廃止されたため、このパラメータは廃止されました。

> **重要:**
> このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.core.brokerCoder.wireFormat
Integration Server が Broker メッセージのエンコードおよびデコードに使用するワイヤリングタイプを指定します。デフォルトは 3 です。

使用可能な設定を以下の表に示します。

設定値	Integration Server
0	タグ情報が無効な場合のみ Broker コンテキストからの現在の参照を削除し、エンコーディングを続行します。
1	タグ情報が有効であるかどうかは検証せず、常に Broker コンテキストからの現在の参照を削除し、エンコーディングを続行します。
2	Broker に固有のデフォルト値を使用して、Broker メッセージをエンコードおよびデコードします。

設定値	Integration Server
	注: 「2」または「3」に設定した場合、Integration Server は「is_Surrogate」フラグを「true」に設定して Broker メッセージのエンコーディングを拡張します。デフォルトでは、「is_Surrogate」が設定されている場合、Broker は BROKER_INT ワイヤリングタイプを使用してエンコードおよびデコードします。これは、Broker のデフォルトです。
3	Broker で設定された現在の値を使用して、Broker メッセージをエンコードおよびデコードします。 **注:** 「2」または「3」に設定した場合、Integration Server は「is_Surrogate」フラグを「true」に設定して Broker メッセージのエンコーディングを拡張します。デフォルトでは、「is_Surrogate」が設定されている場合、Broker は BROKER_INT ワイヤリングタイプを使用してエンコードおよびデコードします。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.core.datatype.patternMatcherPool.delayFill
指定された最小パターン Matcher プールオブジェクトを使用して Integration Server がプールを初期化するかどうかを指定します。「`false`」(デフォルト) に設定した場合、Integration Server はスキーマを解析または検証するときに必ずプールを初期化します。delayFill を「`true`」に設定した場合、Integration Server はプールを初期化しません。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.core.datatype.patternMatcherPool.maxSize
Integration Server がプール内で許容するパターン Matcher プールオブジェクトの最小数を指定します。デフォルトは 50 です。

watt.core.datatype.usejavaregex
オブジェクトの妥当性検査中に、Integration Server で Java 正規表現コンパイラを使用するかどうかを指定します。watt.core.datatype.usejavaregex を「`true`」に設定した場合、XML 妥当性検査中に、Integration Server は Java 正規表現コンパイラを使用し、Integration Server は java.util.regex.pattern で記述されているとおりにパターンマッチングを実行します。「`false`」に設定した場合、XML 妥当性検査中に、Integration Server は Perl 正規表現コンパイラを使用し、パターンマッチングを実行します。デフォルトは「`false`」です。

注:
watt.core.datatype.usejavaregex パラメータは、pub.shema:validate サービスによって実行される妥当性検査と、Integration Server によって実行される Web サービス要求および

応答の妥当性検査を含む、XML 妥当性検査に影響します。watt.core.datatype.usejavaregex parameter はまた、Integration Server が、XML スキーマ定義から IS スキーマまたは IS ドキュメントタイプを作成するときや、スキーマ定義またはスキーマ定義への参照を含む WSDL ドキュメントから Web サービス記述子を作成するときに実行するスキーマの妥当性検査にも影響します。

watt.core.generatedTypeName.namespaceName.authority
サービス URI を作成するために Integration Server が使用するネームスペースエレメントを指定します。ホスト名を示すストリング値を指定してください。デフォルトは「`localhost`」です。

watt.core.generatedTypeName.localName.prefix
タイプ名に Integration Server が使用するプリフィックスを指定します。タイプ名に付加されて、ローカル名を形成します。デフォルトは「__」 (2 つのアンダースコア) です。

watt.core.generatedTypeName.namespaceName.authority
サービス URI を作成するために使用されます。URI の形式は「http://認証パス/サービス名」で、「認証」は、このプロパティで指定される値を表します。通常はホスト名を示すストリング値です。

watt.core.nodeIterator.retainFailures
moving-window モードの NodeIterator がスキップされた XML ノードを保持するかどうかを指定します。watt.core.nodeIterator.retainFailures が「true」に設定されている場合、moving-window モードの NodeIterator 操作は pub.xml:getXMLNodeIterator に指定された条件に一致するまでスキップされた XML ノードを保持します。NodeIterator は、スキップされた XML ノードを pub.xml:getNextXMLNode の成功後にのみ削除します。
watt.core.nodeIterator.retainFailures が「false」に設定されている場合は、moving-window モードの NodeIterator 操作は、pub.xml:getXMLNodeIterator の実行ごとに、スキップされた XML ノードを開放します。デフォルト値は「false」です。

watt.core.schema.createSchema.omitXSDAny
Integration Server で WSDL のスキーマ定義を生成するときに、開いているドキュメントに対応する xsd:any エレメントを複合型定義から省略するかどうかを指定します。[**指定のないフィールドを許可**] プロパティが「true」に設定されている場合、ドキュメント変数は「open」と見なされます。watt.core.schema.createSchema.omitXSDAny を「true」に設定した場合、ドキュメント変数の [**指定のないフィールドを許可**] プロパティが「true」に設定されていても、xsd:any エレメントは WSDL ドキュメントのスキーマ部分から省略されます。
watt.core.schema.createSchema.omitXSDAny を「false」に設定した場合、Integration Server は、対応するドキュメント変数の [**指定のないフィールドを許可**] プロパティが「true」に設定されているときにのみ、xsd:any エレメントを複合型定義に含めます。
watt.core.schema.createSchema.omitXSDAny のデフォルトは「true」です。

注:
個々の Web サービス記述子に対し、[**WSDL から xsd:any を省略する**] プロパティの値は watt.core.createSchema.omitXSDAny サーバ設定パラメータの値を上書きします。[**WSDL から xsd:any を省略する**] プロパティの使用方法の詳細については、*webMethods Service Development Help*を参照してください。

watt.core.schema.anonymousCyclicExtensionDepth

親の複合型拡張を参照する匿名の複合型定義を含む XML スキーマ定義から IS スキーマを作成する場合のネストレベルを指定します。デフォルト値は 3 です。この場合、3 番目のネストレベルで拡張を停止します。この設定は、拡張の階層で、前、現在、または次の参照を処理する必要がある場合に適していることがあります。

watt.core.schema.generateAllTypeDocuments
Integration Server で XML スキーマ定義から IS ドキュメントタイプを生成するときに、すべての複合型の IS ドキュメントタイプをそれらの参照の有無に関係なく生成するかどうかを指定します。このプロパティを「`true`」に設定した場合、Integration Server は、XML スキーマ内の複合型定義ごとに IS ドキュメントタイプを生成します。このプロパティを「`false`」に設定した場合、Integration Server は、複合型が参照されているか、参照されている複合型から派生したものであるときにのみ、その複合型の IS ドキュメントタイプを個別に生成します。デフォルトは「false」です。

注:
このパラメータは無効です。

watt.core.schema.generateObsoleteDocumentTypeNames
グローバルエレメント宣言から参照された複合型のドキュメントタイプを生成するときに、Integration Server がバージョン 8.2 より前に存在した命名規則を使用するかどうかを指定します。ドキュメントタイプを作成するとき、このパラメータが「true」に設定してあり、Designer で [**参照先エレメントのドキュメントタイプを常に生成する**] チェックボックスがオンになっている場合、Integration Server は参照先エレメントに対して作成するドキュメントタイプの名前の先頭に「docType」を使用します。このパラメータを「false」に設定した場合、Integration Server は参照先エレメントに対して作成するドキュメントタイプの名前の先頭に「docTypeRef」を使用します。デフォルトは「false」です。

重要:
Software AG では、Integration Server 8.2 より前に存在した命名規則の使用をお勧めしません。古い命名規則を使用すると、より最近の命名規則に準拠し、その命名規則を想定している新しい機能に支障をきたす可能性があります。

watt.core.schema.generateSubstitutionGroups
置換グループを含む XML スキーマ定義から IS ドキュメントタイプを生成するときに、作成したドキュメントタイプに、置換グループの各メンバーに対応したオプションエレメントを含めるかどうかを指定します。このプロパティを「`false`」に設定した場合、作成されたドキュメントタイプには、置換グループのヘッドエレメントに対応する 1 つのフィールドが含まれますが、置換グループのメンバーに対応するエレメントは含まれません。このプロパティを「`true`」に設定した場合、作成されたドキュメントタイプには、ヘッドエレメントに対応する 1 つのフィールドと、置換グループの各メンバーエレメントに対応するフィールドが含まれます。ヘッドエレメントを含むすべてのフィールドに、オプションエレメントを示すマークが付けられます。デフォルトは「`false`」です。

watt.core.schema.useUnboundedForMaxOccurs
この数より大きい maxOccurs 値が無制限として扱われる数を指定します。maxOccurs が watt.core.schema.useUnboundedForMaxOccurs より大きい要素を持つ IS スキーマに対してドキュメントの妥当性検査を行う場合、Integration Server は、maxOccurs の値が無制限であるように、インスタンスドキュメントの妥当性検査を行います。maxOccurs を無制限として扱うと、9999 など、maxOccurs の値が大きいときに、妥当性検査中に OutOfMemoryError の

発生を防ぐことができます。このパラメータのデフォルト値はありません。つまり、Integration Server ではスキーマに指定された値を maxOccurs として使用します。

注:
このパラメータを使用するには、拡張設定に追加する必要があります。

watt.core.schema.validateIncomingXSD
これは内部プロパティです。変更しないでください。

watt.core.template.enableFilterHtml
Integration Server で、フローサービスおよび %value *Variable*% タグからの出力を、DSP または出力テンプレートで HTML エンコードするかどうかを示します。このプロパティを「true」に設定すると、XML や JavaScript などの %value *Variable*% タグの値が HTML エンコードされます。Integration Server で値を HTML エンコードすると、クロスサイトスクリプティング攻撃を防ぐことができます。このプロパティが「true」の場合でも、encode(none) オプション (%value *Variable* encode(none)%) を含めることにより、Integration Server で %value *Variable*% タグの出力を HTML エンコードしないでおくことができます。このプロパティを「false」に設定すると、%value *Variable*% タグの値はエンコードされません。このため、DSP ページまたは出力テンプレートから返されるページは、クロスサイトスクリプティング攻撃に対して脆弱になることがあります。フローサービスの場合、このプロパティが「true」に設定されていると、フローサービスからの出力 (「<」、「>」など) はエンコードされます。このプロパティを「false」に設定すると、フローサービス出力のエンコーディングは無効になります。デフォルトは「true」です。

watt.core.template.enableSecureUrlRedirection
DSP ページの場合、このパラメータは Integration Server がセキュア URL リダイレクトを実行するかどうかを制御します。通常、内部 DSP ページでは、変数名として「url」や「returnurl」を %value% タグと共に使用することにより、クライアントで URL リダイレクトを実行可能であることを Integration Server に示します。このため、DSP ページが内部 DSP ページかカスタム DSP ページかに関係なく、Integration Server で %value% タグに変数名「url」または「returnurl」が検出されると、この値は URL リダイレクト用であると見なされます。

watt.core.template.enableSecureUrlRedirection パラメータを「true」に設定すると、Integration Server はセキュア URL リダイレクトを使用します。セキュア URL リダイレクトを実行するときに、%value% タグに変数名「url」または「returnurl」を使用すると、Integration Server は URL をチェックして、それが Integration Server 内の場所への相対パスか (..\redirectedurl.dsp など)、または別の値かを判別します。相対パスの場合、Integration Server はその URL へのリダイレクトが安全であると見なすため、何も行いません。ただし、他の値の場合、Integration Server は URL が外部の URL (http://example.com など) であると見なし、%value% タグがその URL へのリダイレクトを意図していない場合でも、出力を変更します。Integration Server は URL を変更して、外部 URL へのリダイレクトが試行されたときに、エラーページにリダイレクトされるようにします。URL の変更は、%value% タグの出力の先頭に値「error.dsp?data=」を追加することによって行います (error.dsp?data=http://example.com など)。

このパラメータを「false」に設定すると、セキュア URL リダイレクトが無効になるため、アプリケーションが危険にさらされる可能性があります。

内部 URL にリダイレクトする場合にのみ、%value% タグに変数名「url」または「returnurl」を使用します。変数名「url」または「returnurl」を使用する既存のアプリケーションがある

場合、アプリケーションを更新し、内部 URL リダイレクト以外に対してはこれらの変数名を使用しないことをお勧めします。既存のアプリケーション内の変数名を変更しない場合は、watt.core.template.enableSecureUrlRedirection パラメータを「false」に設定します。

watt.core.template.enableSecureUrlRedirection パラメータのデフォルト値は「true」です。

watt.core.transientStore.logExceptions

Integration Server がら例外を一時ストアからログに記録するかどうかを指定します。中断、デッドロックなど、一時ストアによってスローされた一部の例外条件は内部的に処理され、Integration Server は暗黙のうちに例外から回復します。「true」に設定した場合、Integration Server は、コンシューマスレッドを終了させる可能性がある、一時ストアからスローされた例外をログに記録します。これは、再サブミットプロセス中に内部ストアで問題を診断するときに役立つことがあります。「false」に設定した場合、Integration Server は一時ストアからのすべての例外をログを記録しません。デフォルトでは、Integration Server が処理して回復できる例外からの不必要な警告を避けるため、このパラメータは false に設定されます。

Software AG では、Software AG Global Support によって指示された場合にのみ、このパラメータを「true」にすることをお勧めします。

watt.core.validation.skipAbsentStarBody

列挙制限がある混合コンテンツエレメントをデコードするときに Integration Server が実行する妥当性検査をスキップするかどうかを指定します。オプションのコンテンツがなかった場合、妥当性検査でエラーが返されました。これは、空の値が *body フィールドに設定された列挙のオプションとしてリストされていなかったためです。「true」に設定された場合、Integration Server はこの妥当性検査をスキップし、*body を完全にオプションとして扱うため、*body の値が空であるか存在しない場合も妥当性検査に合格します。

「false」に設定した場合、Integration Server は妥当性検査を実行します。デフォルト値は「false」です。

このプロパティの変更は即座に反映されます。

watt.core.validation.multipleroot

マルチパートドキュメントを処理するときに pub.schema:validate サービスで複数のルートの妥当性検査を行うかどうかを指定します。このプロパティを「true」に設定した場合、pub.schema:validate サービスは複数のルートノードをチェックします。複数のルートノードが検出された場合、妥当性検査エラーとなります。このプロパティを「false」に設定した場合、pub.schema:validate サービスは複数のルートの妥当性検査を行いません。デフォルトは「true」です。

watt.core.validation.skipMandatoryFields

ドキュメントの妥当性検査で必須フィールドがドキュメントにないことがわかった場合に Integration Server でエラーを生成するかどうかを指定します。

watt.core.validation.skipMandatoryFields を「true」に設定した場合、Integration Server は、ドキュメントの妥当性検査時に必須フィールドがドキュメントになくても、妥当性検査エラーを生成しません。このパラメータを「false」に設定した場合、Integration Server は、必須フィールドがドキュメントになければ、妥当性検査エラーを生成します。デフォルトは「false」です。このパラメータが影響するのは、pub.schema:validate サービスによって実行されるドキュメントの妥当性検査と SOAP 要求および SOAP 応答の妥当性検査です。

watt.core.validation.skipNoNamespaceReference

Integration Server でスキーマに対する XML 妥当性検査中に、名前空間で完全修飾された要素内から作成された名前空間で完全修飾されていない要素へ参照の妥当性検査をスキップす

るかどうかを示します。Integration Server でこれらの参照要素の妥当性検査をスキップする場合は、`true` に設定します。Integration Server でこれらの参照要素の妥当性検査を実行する場合は、`false` に設定します。デフォルトは「`false`」です。

watt.core.validation.w3cConformant
W3C 勧告の『XML Schema Part 2: Datatypes』に従って XML の妥当性を検査する場合に、データタイプのインスタンスに不正な値があるかどうかを Integration Server が評価するかどうかを示します。Integration Server でデータタイプが W3C 勧告と一致することを検査する場合は、watt.core.validation.w3cConformant を「`true`」に設定します。デフォルトは「`false`」です。

watt.core.xml.allowedExternalEntities
XML ファイルまたは XSLT ファイルに変換する XML コンテンツを含む信用のある外部エンティティ (ファイル URI、HTTP URL など) のリストを指定します。カンマを使用して各エンティティを区切ります。

pub.xslt.Transformations:transformSerialXML サービスは、このリスト内の任意のエンティティのコンテンツをロードします。リストは、サービスが受信する XML で参照されるか、サービスが XML を変換するために使用する XSLT スタイルシートで参照されます。サービスは、以下の状況でこのリストを読み取ります。

- サービスの loadExternalEntities 入力パラメータが「`false`」に設定されている。
- サービスの loadExternalEntities 入力パラメータがサービスの署名で指定されておらず、watt.core.xml.expandGeneralEntities サーバパラメータが「`false`」に設定されている。

watt.core.xml.expandGeneralEntities
pub.xml:loadXMLNode サービスおよび pub.xml:xmlStringToXMLNode サービスが、拡張一般 (内部および外部) エンティティを返すかどうかを示します。また、pub.xslt.Transformations:transformSerialXML サービスが外部エンティティを許可するかどうかも示します。このパラメータが「`true`」に設定されると、loadXMLNode および xmlStringToXMLNode サービスは一般エンティティへの参照を拡張し、transformSerialXML サービスは外部エンティティを許可します。このパラメータが「`false`」に設定されると、loadXMLNode および xmlStringToXMLNode サービスは一般エンティティへの参照を無視し、transformSerialXML サービスは外部エンティティをブロックします。デフォルトは「`true`」です。

> **注:**
> このパラメータは、loadXMLNode、xmlStringToXMLNodeの各サービスおよび任意の XSLT サービスのすべてのインスタンスに適用されます。ただし、expandGeneralEntities 入力パラメータが loadXMLNode または xmlStringToXMLNode サービスに指定されている場合や loadExternalEntities パラメータが transformSerialXML サービスに指定されている場合、サービスレベルの設定が watt.core.xml.expandGeneralEntities よりも優先されます。

watt.core.xsd.useGeneratedURIForCreateXSD
XSD の作成時に Integration Server が URI にノードのパス、ノードとスキーマにネームスペース名を使用するかどうかを指定します。「`true`」に設定すると、Integration Server はノードのパスに基づいて URI を生成します。この URI は特定のノードまたはスキーマのネームスペース名として使用されます。「`false`」 (デフォルト) に設定した場合、Integration Server はネームスペースを NULL に設定します。

watt.core.xsd.useGeneratedURIForNonRPC
Integration Server がメッセージに対して生成される URI を WSDL メッセージのネームスペース名として使用するかどうかを指定します。「`true`」(デフォルト) に設定すると、Integration Server は形式が非 RPC の場合に限り、メッセージに対して生成される URI を WSDL メッセージのネームスペース名として使用します。「`false`」に設定した場合、Integration Server はネームスペースを NULL に設定します。

watt.core.xsd.useKnownSchemaLocation
プロバイダ Web サービス記述子用に生成された WSDL ドキュメントの import ステートメントに対する登録済みスキーマの場所と schemaLocation 属性を Integration Server に含めるかどうかを指定します。schemaLocation 属性が WSDL ドキュメント内にない場合、WSDL ドキュメントを使用する Integration Server は Web サービス処理中に schemaLocation 属性値を逆参照しません。登録済みスキーマの場所がネームスペースの解決に必要だった唯一の import ステートメントである場合、schemaLocation 属性がないと、インターネットに接続されていない間に Web サービス開発が可能になります。ネームスペースが次のいずれか 1 つである場合に、import ステートメントで schemaLocation 属性を省略するには、このパラメータを「false」に設定します。

http://www.w3.org/XML/1998/namespace

http://www.w3.org/2003/05/soap-envelope

デフォルト値は true です。

> **注:**
> メモ: WS-I 準拠では、WSDL ドキュメントまたは WSDL ドキュメントで参照されるファイルですべての XML スキーマインポートに schemaLocation 属性が必要です。Web サービス記述子用に生成される WSDL の XML スキーマに、指定されたネームスペースのいずれか用の import ステートメントが含まれる場合、watt.core.xsd.useKnownSchemaLocation を「false」に設定すると、WS-I 準拠が解除されます。

watt.debug.

watt.debug.layout
サーバのログファイルに記録するメッセージおよび [**ログ**] > [サーバ] ページに表示するメッセージの形式を指定します。次のいずれかの形式を指定できます。

- `new`

  次のメッセージ形式が使用されます。

  (*Component*) [*ComponentID*.00*SubComponentID.SubComponentID.MessageKey*] *TimeStampMessageTypeMessageText*

  (IS.SERVER) [ISS.0025.25.6] 2007-07-31 10:45:27 EDT INFO: License Manager が起動しました

- `legacy`

  この形式は、バージョン 7.1 よりも前の Integration Server で使用されていたメッセージ形式に対応します。以前のメッセージ形式との下位互換性を維持する必要がある場合は、この形式を使用してください。たとえば、サーバログに書き込まれたメッセージの処理用のコードを作成してある場合などが該当します。

メッセージ形式として「legacy」を選択した場合、メッセージは次の形式で表示されます。

TimeStamp [*ComponentID*.00*SubComponentID.MessageKeyMessageType*] *MessageText*

2007-07-31 10:39:59 EDT [ISS.0025.0006I] License Manager が起動しました

これがデフォルトです。

watt.debug.level
サーバのログファイルに記録するデバッグ情報および [**ログ**] > [サーバ] ページに表示するデバッグ情報のレベルを指定します。デフォルトは「Info」です。

指定内容	表示内容
Off	メッセージを表示しない
Fatal	重大メッセージのみ
Error	エラーメッセージおよび重大メッセージ
Warn	警告メッセージ、エラーメッセージ、および重大メッセージ
Info	情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージこれがデフォルトです。
Debug	デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ
Trace	トレースメッセージ、デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージ、および重大メッセージ

注:
[**ログ**] > [ログ設定] > [サーバロガー] ページでデフォルト機能のログレベルを設定して、watt.debug.level プロパティの値を設定する方法もあります。ログの設定の詳細については、237 ページの「サーバログに記録する情報の量と種類の指定」を参照してください。

Integration Server 7.1 よりも前の Integration Server では、数値ベースのシステムを使用してサーバログに記録するデバッグ情報のレベルを設定していました。Integration Server では、このシステムとの下位互換性が維持されています。次の表は、数値ベースのシステムを示しています。

指定内容	記録されるメッセージ
0	重大メッセージのみ
1	エラーメッセージおよび重大メッセージ
2	警告メッセージ、エラーメッセージおよび重大メッセージ
3、4	情報メッセージ、警告メッセージ、エラーメッセージおよび重大メッセージ

指定内容	記録されるメッセージ
5、6、7	デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージおよび重大メッセージこれがデフォルトです。
8、9、10	トレースメッセージ、デバッグメッセージ、情報メッセージ、警告メッセージ、エラーメッセージおよび重大メッセージ 詳細レベル値を高く設定すると、サーバで記録される情報メッセージの詳細レベルが高くなります。

watt.debug.logfile
Integration Server でサーバログ情報を書き込む先のファイルへの完全修飾パスまたは相対パスを指定します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* \instances*instance_name* に相対的なパスです。

ディレクトリとファイル名を指定する必要があります。

デフォルトは *Integration Server_directory* \instances*instance_name*\logs\server.log です。

注:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.debug.warnOnClasspathError
見つからないクラスパスエントリに関する警告メッセージを標準出力に書き込むかどうかを指定します。このパラメータを「`true`」に設定した場合、メッセージ「ini.cnf のクラスパスエントリが見つかりません: <*classpath_entry_name*>」が標準出力に書き込まれます。このメッセージが書き込まれないようにする場合は、このパラメータを「`false`」に設定します。デフォルトは「`false`」です。

このメッセージは Integration Server ログ機能が初期化される前に生成されるため、警告メッセージが標準出力に書き込まれます。

この設定パラメータは、[拡張設定] または server.cnf には表示されません。Integration Server でこれらの警告を書き込むには、このプロパティをコマンドラインオプションとして Integration Server に渡す必要があります。以下のような行を *Software AG_directory* /profiles/IS_instance/configuration/custom_wrapper.conf に追加します。

```
wrapper.java.additional.NNN=-Dwatt.debug.warnOnClasspathError=true
```

ここで *NNN* は次に使用可能な wrapper.java.additional 番号です。

Microservices Runtime の実行時の警告を書き込むには、テキストエディタを使用して、*Integration Server_directory* /bin/server.bat(sh) の JAVA_CUSTOM_OPTS パラメータのプロパティを指定します。次に例を示します。

```
set JAVA_CUSTOM_OPTS="-Dwatt.debug.warnOnClasspathError=true"
```

watt.debug2.

watt.debug2.facList
サーバが情報をログに取得する対象となる、カンマで区切られた有効な機能のリストを指定します。機能には番号が付けられています。デフォルトは `999` で、サーバがすべての機能の

情報をログに記録することを示します。1000 を指定すると、サーバはいずれのサービスの情報もログに記録しません。

機能の名前を表示するには、サーバが情報を記録する対象の機能を有効または無効にする Integration Server Administrator の [**設定**] > [ログ] ページを使用します。

watt.debug2.logstringfile
エラーコードおよび機能を含むディクショナリファイルの名前 (拡張子 .txt なし) を指定します。デフォルトは「lib\logstr」 (英語バージョン) です。

watt.frag.

watt.frag.keep.original.servicename
Java サービスのフラグメント化またはコンパイル時に Integration Server が入力フィールドの元のサービス名を保持するかどうかを指定します。このパラメータを「true」に設定した場合、Integration Server は元のサービス名を保持します。「false」に設定した場合、Integration Server は元のサービス名を変更します。デフォルトは「false」です。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.infradc.

watt.infradc.artmonitor
これは内部パラメータです。変更しないでください。

watt.infradc.artpollinterval
これは内部パラメータです。変更しないでください。

watt.net.

watt.net.clientKeepaliveAgingLimit
ソケットのキープアライブ時間 (秒単位) を指定します。プールにソケットを返す前に、Integration Server はソケット接続の経過時間を watt.net.clientKeepaliveAgingLimit の値と比較します。ソケットの経過時間が watt.net.clientKeepaliveAgingLimit の値より長い場合、Integration Server はソケットをプールに返しません。代わりに、Integration Server はソケットを閉じます。別のソケット接続が必要な場合、Integration Server はソケット接続を作成します。ソケットの経過時間が watt.net.clientKeepaliveAgingLimit の値より短い場合、Integration Server はソケットを接続プールに返します。デフォルトは 180 秒です。

注:
接続の経過時間が watt.net.clientKeepaliveAgingLimit パラメータより短い場合でも、接続が watt.net.clientKeepaliveUsageLimit サーバ設定パラメータで設定された使用上限を超えている場合、Integration Server は接続を閉じます。

注:
watt.net.maxClientKeepaliveConns が 0 より大きい値に設定されている場合にのみ、watt.net.clientKeepaliveAgingLimit パラメータは適用されます。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.clientKeepaliveTimeout
クライアントのキープアライブ接続がアイドル状態になってから Integration Server によって終了されるまでの期間 (秒数) を制御します。デフォルトは 180 秒 (3 分) です。

watt.net.clientKeepaliveUsageLimit
クライアント接続プールにおけるソケットの最大使用数を指定します。ソケットをプールに返す前に、Integration Server はソケットが要求の送信に使用された回数を watt.net.clientKeepaliveUsageLimit の値と比較します。ソケットの使用回数の方が watt.net.clientKeepaliveUsageLimit の値より多い場合、Integration Server はソケットをプールに返しません。代わりに、Integration Server はソケットを閉じます。プールで新しいソケットが必要な場合、Integration Server はソケットを作成します。デフォルト値は 100 回です。

注:
接続の使用回数が watt.net.clientKeepaliveUsageLimit パラメータより少ない場合でも、接続が watt.net.clientKeepaliveAgingLimit サーバ設定パラメータで設定された経過時間の上限を超えている場合、Integration Server は接続を閉じます。

注:
watt.net.maxClientKeepaliveConns が 0 より大きい値に設定されている場合にのみ、watt.net.clientKeepaliveUsageLimit パラメータは適用されます。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.default.accept
pub.client:http サービスの *headers* 入力パラメータに Accept ヘッダーがない場合、Accept ヘッダーのデフォルト値を指定します。watt.net.default.accept に値がなく、Accept ヘッダーが *headers* パラメータにない場合、pub.client:http サービスが送信する要求に Accept ヘッダーは含まれません。デフォルトで、watt.net.default.accept パラメータに値はありません。

watt.net.email.validateHost
Integration Server が電子メールリスナーの IP アクセス制限を実行するかどうかを指定します。電子メールのポートを定義するときに、電子メールのポートを経由したアクセスを許可または拒否するホストを指定する IP アクセス制限を定義することができます。サーバが電子メールリスナーの IP アドレス制限を実行するようにする場合は、このプロパティを「`true`」に設定し、実行しないようにする場合は「`false`」に設定します。デフォルトは「`true`」です。

watt.net.encodeToUpperCase
Integration Server が要求 URL 内の文字をエンコードするときに大文字を使用するかどうかを指定します。Integration Server は、URL 要求を処理するときに、「%」とそれに続く 2 桁の 16 進数値を追加することで、URL 内で ASCII セット外の文字をエンコード文字に変換します。このパラメータを「`true`」に設定した場合、Integration Server は 16 進数値で大文字を使用します。たとえば、「value1>4」のエンコード値は「value1%3E4」です。「`false`」に

設定した場合、Integration Server は文字を大文字に変換しません。たとえば、「value1>4」のエンコード値は「value1%3e4」です。デフォルトは「`true`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ftp.ignoreErrors
カンマ区切りのリストを使用して、FTP クライアントで無視する FTP コマンドエラーコードを指定します。たとえば、このプロパティを「501, 505」に設定すると、FTP クライアントはエラーコード 501 および 505 を無視します。

watt.net.ftp.noExtensionKey
入力ファイルに拡張子が付いていない場合に、ファイルの MIME タイプを決定するために Integration Server が使用する拡張子を指定します。FTP コマンド経由で Integration Server サービスを呼び出す場合に、Integration Server は指定された拡張子を使用します。デフォルトは「`ftp_no_extension`」で、この場合、拡張子がないために Integration Server は MIME タイプを決定できません。

watt.net.ftp.securitychecks
FTP クライアントが Integration Server FTP ポートにアクセスするときに、Integration Server が FTP セキュリティチェックを実行するかどうかを指定します。Integration Server に FTP 関連のセキュリティチェックを実行させる場合は、このプロパティを「`true`」に設定します。Integration Server に FTP 関連のセキュリティチェックを実行させない場合は、このプロパティを「`false`」に設定します。デフォルトは「`true`」です。

watt.net.ftpClientDataConnTimeout
アクティブモードで実行されている組み込み FTP サービス (*transfertype* 入力パラメータで指定) にリモートの FTP サーバが接続するときの許容待機時間をミリ秒単位で指定します。指定された時間内に接続が確立されない場合は、例外がスローされます。デフォルト値は 30000 ミリ秒 (30 秒) です。

watt.net.ftpClientTimeout
FTP セッションがアイドルでいられる時間を秒単位で指定します。この時間を過ぎると、FTP セッションはメモリから消去されます。デフォルトは 600 秒 (10 分) です。

watt.net.ftpConnTimeout
FTP リスナーがクライアントとの非アクティブな接続の存続を許容する最大時間をミリ秒単位で指定します。デフォルトは 15 分です。

watt.net.ftpDataConn
FTP サーバとして機能している Integration Server で、複数の同時接続を許可し、並列ダウンロードをサポートするかどうかを指定します。このパラメータを「true」に設定した場合、Integration Server は並列ダウンロードを許可し、同じ FTP セッションを再使用します。このパラメータを「false」に設定した場合、Integration Server は並列ダウンロードを許可せず、同じ FTP セッションを再使用します。デフォルトは「false」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ftpDataConnTimeout

ファイルアップロードの実行時に FTP リスナーが次の読み取りまで待機する最大時間をミリ秒単位で指定します。デフォルトは 60000 ミリ秒 (60 秒) です。

watt.net.ftpPassiveLocalAddr
PORT コマンドによって送信されるアドレスを指定します。ホスト名または IP アドレスを指定できます。

> **注:**
> このパラメータは、FTP/FTPS ポートが IPv6 アドレスにバインドされている場合は適用されません。その場合、パッシブモードリスンアドレスはポートバインドアドレスと同じです。

パッシブモードで実行した場合、FTP または FTPS ポートは FTP または FTPS クライアントに PORT コマンドを送信します。PORT コマンドは、クライアントがデータ接続を作成するための接続先アドレスおよびポートを指定します。ただし、FTP または FTPS ポートが NAT サーバの内側にある場合、Integration Server が実行されているホストのアドレスは FTP または FTPS クライアントからは見えません。このため、PORT コマンドには、クライアントがサーバへの接続に必要とする情報が含まれません。watt.net.ftpPassiveLocalAddr プロパティの値を指定することで、この問題を解決できます。

また、FTP または FTPS ポートを設定する際に (200 ページの「FTP ポートの追加」または 194 ページの「FTPS ポートの追加」を参照)、[**パッシブモードリスンアドレス**] フィールドを使用して、個々の FTP または FTPS ポートのパッシブモードアドレスを指定することもできます。この方法を使用すれば、FTP ポートごとに異なるパッシブモードアドレスを指定できます。[**パッシブモードリスンアドレス**] フィールドと watt.net.ftpPassiveLocalAddr プロパティの両方にアドレスが指定されている場合、PORT コマンドは、watt.net.ftpPassiveLocalAddr プロパティに指定された値を使用します。

watt.net.ftpPassivePort.max
受信転送モード (PASV) を使用するクライアントのデータ接続に使用する FTP/FTPS リスナーのポート番号範囲の最大ポート番号を指定します。watt.ftpPassivePort.min と併せて使用する必要があります。使用方法については、watt.ftpPassivePort.min の説明を参照してください。

watt.net.ftpPassivePort.min
受信転送モード (PASV) を使用するクライアントのデータ接続に使用する FTP/FTPS リスナーのポート番号範囲の最小ポート番号を指定します。watt.ftpPassivePort.max と併せて使用する必要があります。これらのプロパティを使用してポート範囲を指定した場合、指定された最小値と最大値の範囲内のポートのみが、受信 FTP/FTPS クライアントデータ接続の受信待機ポートとして使用されます。ファイアウォールの管理者は指定されたポートを開いておくだけです。

操作上の考慮事項

- 両方のプロパティが存在しない、または未定義の場合、FTP/FTPS リスナーは、任意のフリーポート上で受信待機するという従来の動作を維持します。
- watt.net.ftpPassivePort.min に指定された値が 1 未満のときには、1 というデフォルト値が使用されます。watt.net.ftpPassivePort.max に指定された値が 65534 を超えるときには、65534 というデフォルト値が使用されます。これら両方の条件が同時に存在する場合、FTP/FTPS リスナーは、任意のフリーポート上で受信待機するという従来の動作を維持します。
- 指定された値が予想された範囲内にない場合は、コマンドチャネルから FTP/FTPS クライアントにエラーメッセージが戻されます。たとえば、一方のプロパティが未定義の場

合や、watt.net.ftpPassivePort.min の値が watt.net.ftpPassivePort.max の値よりも大きい場合、または一方のプロパティの数値が無効な場合などがこれにあたります。

- また、指定されたポート範囲のすべてのポートが使用中の場合も、エラーメッセージが戻されます。
- 個々のエラーメッセージの詳細については、serverYYYYMMDD.log ファイルに記録されます。

これらの設定値の定義後に Integration Server を再起動する必要はありません。ポート範囲のプロパティは Integration Server Administrator を使用していつでも変更できます。

watt.net.ftpSweepInterval
FTP スイーパの実行間隔を秒単位で指定します。FTP スイーパはメモリ内の FTP セッションを定期的に走査し、設定されたアイドル時間を過ぎたセッションを消去します。FTP スイーパはデフォルトでは 600 秒 (10 分) ごとに実行されます。

watt.net.ftpUseCertMap
FTPS ポートで受信した要求に関して、Integration Server が認証マップを尊重するかどうかを指定します。

このプロパティを「false」に設定すると (デフォルト設定)、Integration Server はクライアントの認証に指定されたユーザを無視して、ユーザ ID/パスワードのプロンプトを通じて入力された情報を使用してユーザをログインします。

このプロパティを「`true`」に設定した場合、クライアントの認証が以前に Integration Server ユーザにマッピングされていれば、Integration Server はクライアントの認証に指定されたユーザ ID としてユーザをログインします。Integration Server では、ユーザ ID/パスワードのプロンプトから入力されたユーザ ID は無視されます。

たとえば、watt.net.ftpUseCertMap が「false」に設定されていて、認証が以前にユーザ「Alice」にマッピングされていたとします。ユーザ「Alice」の認証を提示し、Alice のユーザ名とパスワードをプロンプトに入力したユーザは、Integration Server に Alice としてログインすることになります。しかし、同じ認証を使用した場合でも、Bob のユーザ名とパスワードをプロンプトに入力したユーザは、Integration Server に Bob としてログインします。つまり、Integration Server では認証マップは無視されます。

> **注:**
> [FTPS リスナーの設定] ページのクライアント認証の設定値 [**None**]、[**クライアント認証を要求する**] および [**クライアント認証を必須にする**] は、Integration Server が認証を要求するかどうか、および認証が提示されない場合にIntegration Server がどのように動作するかを制御します。watt.net.ftpUseCertMap プロパティは、FTP クライアントから認証が提示された場合の Integration Server の動作を制御します。FTPS ポートおよび HTTPS ポートでのクライアント認証の詳細については、552 ページの「クライアント認証による認証」を参照してください。認証のマッピングの詳細については、553 ページの「クライアント認証のインポートとユーザへのマッピング」を参照してください。

watt.net.ftpUseDefaultContentHdlr
認知されていないファイル拡張子による受信要求を Integration Server の FTP 待機ポートで処理する方法を指定します。「`true`」に設定すると、FTP 待機ポートは、コンテンツを text/html として扱うデフォルトのコンテンツハンドラを使用して受信要求を処理します。「`false`」に設定すると、FTP 待機ポートは認知されていないファイル拡張子を使用した受信要求を受信したときに、例外を返します。デフォルトは「`true`」です。

watt.net.http.clientSession.idleTimeout
新規作成された送信 HTTP/S セッションがタイムアウトになり、セッションスイーパによって削除される前に、そのセッションがアイドル状態でいることができる時間をミリ秒単位で指定します。セッションのアイドルタイムアウト時間を少なくすると、Integration Server は再使用される可能性が少ないセッションを早めに削除できます。デフォルト値は 20,000 ミリ秒 (20 秒) です。送信 HTTP/S セッションの名前は (httpclient) です。

watt.net.http.followRedirect
送信 HTTP 要求が 300～400 の応答コードを受信するときに Integration Server が自動的にリダイレクトに従うかどうかを指定します。「true」に設定すると、送信要求がリダイレクトに従います。「false」に設定すると、送信要求はリダイレクトに従わず、応答コードおよび元の送信要求からのすべての応答ヘッダーを返します。デフォルトは「true」です。

watt.net.http.followRedirect の動作は、pub.client:http サービスの *followRedirect* 入力パラメータによって、個別に上書きできます。

watt.net.http.redirect.performSSRFcheck
Integration Server がリダイレクト要求に従うときに、IP アドレスとプロトコルをチェックして SSRF (Server-Side Request Forgery) を防ぐかどうかを指定します。

「true」に設定した場合、リダイレクト要求に従うことができる送信要求を処理するときに、Integration Server は次の 2 つのチェックを行い、SSRF を防ぎます。

- リダイレクトされた URL では、HTTP または HTTPS プロトコルを使用する必要がある・Integration Server は他のプロトコルのリダイレクトを拒否する。
- リダイレクトされる URL の IP アドレスをプライベートアドレスにすることはできません。

リダイレクトされる URL で HTTP または HTTPS 以外のプロトコルを指定したり、プライベート IP アドレスを指定すると、Integration Server は送信サービスの呼び出し側に汎用エラーを返し、SSRF チェックに関するエラーメッセージおよび拒否の理由をログに記録します。

「false」に設定した場合、リダイレクトに従う前に Integration Server は SSRF チェックを実行しません。

デフォルトは「false」です。

watt.net.httpChunkSize
「Transfer-Encoding: Chunked」を使用して HTTP 要求または応答を送信する場合のデフォルトチャンクサイズを設定します。デフォルトチャンクサイズは 8192 バイトです。最小チャンクサイズは 500 バイトです。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.httpPass
Integration Server がサービスをクライアントとして呼び出すときに使用しなければならないデフォルト HTTP パスワード。

watt.net.httpUser
クライアントとして動作する Integration Server がサービスを呼び出すために使用するデフォルトの認証 HTTP ユーザ名を指定します。たとえば、Integration Server が pub.client:http サー

ビスを呼び出すときに *auth/user* パラメータを指定しないと、Integration Server はこのプロパティの値をユーザ名として使用します。デフォルト値はありません。

watt.net.http401.throwException
pub.client:http サービスが 401 エラー応答を受信したときに、NetException をスローするか、またはパイプラインに HTTP 応答 *header* と *body* を配置するかのいずれかを指定します。watt.net.http401.throwException を「true」に設定すると、pub.client:http サービスは 401 エラーを受信したときに NetException エラーをスローします。watt.net.http401.throwException を「false」に設定すると、pub.client:http サービスは 401 エラーを受信したときに NetException を抑制し、存在する場合は、HTTP 応答ヘッダーと本文をサービス出力の *header* および *body* フィールドに配置します。デフォルトは「true」です。

watt.net.http501-599.throwException
pub.client:http サービスがリモート HTTP サーバから 501～599 レベルの応答を受信した場合に、ServiceException をスローするか、または応答ヘッダーと応答本文を返すかのいずれかを指定します。「true」に設定すると、pub.client:http サービスはリモート HTTP サーバから 501～599 レベルの応答を受信すると、ServiceException をスローします。「false」に設定すると、pub.client:http サービスはリモート HTTP サーバから 501～599 レベルの応答を受信しても、ServiceException をスローしません。代わりに、pub.client:http サービスは、サービスの出力で 501～599 の範囲のステータスコード、応答ヘッダー、応答本文を返します。デフォルトは「true」です。

> **注:**
> リモート HTTP サーバが応答コード 500 を返すと、pub.client:http サービスはステータスコード、応答ヘッダー、応答本文を返します。

watt.net.jsse.client.disabledProtocols
Integration Server が送信要求を作成する SSL クライアントとして動作するときに、無効な SSL とTLS プロトコルバージョンのリストを指定します。デフォルトの無効なプロトコルは `SSLv2Hello,SSLv3,TLSv1,TLSv1.1` です。

このパラメータを変更すると、次に送信接続が作成されたときに変更が反映されます。

watt.net.jsse.client.disabledProtocols プロパティの値は、SSL に JSSE を使用する Integration Server からのすべての送信接続に影響します。

watt.net.jsse.client.enabledCipherSuiteList
カンマ区切りのリストまたはファイルを使用して、送信 HTTPS または FTPS 要求の作成時に使用される JSSE ソケットで使用される暗号スイートを指定します。JVM によってサポートされるすべての暗号スイートを含めるには、このパラメータを `default` に設定します。

次に例を示します。

```
watt.net.jsse.client.enabledCipherSuiteList=TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256

watt.net.jsse.client.enabledCipherSuiteList=default
```

デフォルト値は default です。

このパラメータの値はカンマ区切りリスト、デフォルト、またはファイルへの絶対パスに設定できます。上記の組み合わせを指定することはできません。このパラメータの値としてファイルを指定する方法の詳細については、488 ページの「SSL と使用する暗号スイートの指定」を参照してください。

注:
watt.net.jsse.client.enabledCipherSuiteList プロパティの値は、すべての HTTPS、FTPS、および 電子メールポートと SSL に JSSE を使用する Integration Server の SMTP 送信接続に影響します。

注:
watt.net.jsse.client.enabledCipherSuiteList への変更は、新しい接続のみに影響があります。

watt.net.jsse.server.disabledProtocols
Integration Server が受信要求を処理する SSL サーバとして動作するときに、無効な SSL と TLS プロトコルバージョンのリストを指定します。デフォルトの無効なプロトコルは `SSLv2Hello,SSLv3,TLSv1,TLSv1.1` です。

ポートへのこのプロパティの変更はポートの再起動後に反映されます。ポートを無効にしてから有効にすることによりポートを再起動できます。また、Integration Server を開始時に有効な各ポートは開始されます。

注:
ネゴシエーション中のプロトコルのダウングレードを避けるには、TLSv1.2 以降の 1 つまたは 2 つのプロトコル以外のすべてのプロトコルバージョンを無効になるように watt.net.jsse.server.disabledProtocols を設定します。

注:
ポートごとにプロトコルを無効にすることができます。ポートごとに無効にしたプロトコルは watt.net.jssse.server.disabledProtocols パラメータの設定よりも優先します。ポートごとにプロトコルを無効にするための詳細な情報は、231 ページの「ポートごとに JSSE 用のプロトコルを無効にする」を参照してください。

watt.net.jsse.server.enabledCipherSuiteList
カンマ区切りのリストまたはファイルを使用して、JSSE を使用して受信要求を処理する Integration Server ポートで使用される暗号スイートを指定します。

JVM によってサポートされるすべての暗号スイートを含めるには、このパラメータを `default` に設定します。

次に例を示します。

```
watt.net.jsse.server.enabledCipherSuiteList=TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256

watt.net.jsse.server.enabledCipherSuiteList=default
```

デフォルト値は default です。

このパラメータの値はカンマ区切りリスト、デフォルト、またはファイルへの絶対パスに設定できます。上記の組み合わせを指定することはできません。このパラメータの値としてファイルを指定する方法の詳細については、488 ページの「SSL と使用する暗号スイートの指定」を参照してください。

重要:
このプロパティの変更を有効にするには、ポートを起動する必要があります。ポートが既に起動している場合は、ポートを無効にしてから再起動して有効にできます。

watt.net.jsse.server.SSLSessionTimeout

タイムアウトして SSL キャッシュから SSL セッションを削除するまでに Integration Server が待機する時間 (秒) を指定します。watt.net.jsse.server.SSLSessionTimeout の値は、0 以上である必要があります。値が 0 に設定された場合、SSL セッションはタイムアウトしません。デフォルトではパラメータは空であり、これは Integration Server が JSSE でサポートされているデフォルトのキャッシュ動作に従うことを示します。

注:
このサーバ設定パラメータは JSSE を使用するポートのみに適用されます。

このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.jsse.server.useCipherSuitesOrder
Integration Server が SSL/TLS サーバとして機能し、JSSE を使用する場合、ローカルの暗号スイート設定が SSL/TLS ハンドシェーク中に Integration Server によって尊重される必要があるかどうかを指定します。「true」に設定した場合、Integration Server は SSL/TLS ハンドシェーク中に watt.net.jsse.server.enabledCipherSuiteList で設定された暗号スイートを使用します。watt.net.jsse.server.enabledCipherSuiteList が「default」に設定されているか、空である場合、Integration Server はこのパラメータを無視します。「false」に設定した場合、Integration Server は暗号スイートの順序を無視します。このパラメータのデフォルト値は「false」です。

重要:
このプロパティの変更を有効にするには、ポートを起動する必要があります。ポートが既に起動している場合は、ポートを無効にしてから再起動して有効にできます。

watt.net.localhost
Integration Server を使用しているマシンのホスト名を設定します。デフォルトはありません。

Integration Server が他の Integration Server に自身を認識させる必要がある場合など、Integration Server のホストマシンの IP アドレスが必要になることがあります。この場合、Integration Server では、指定されたホスト名に対して逆 DNS 検索を実行し、ループバックアドレス (IPv4 では 127.0.0.1 または IPv6 では ::1) の代わりにマシンの IP アドレスを提供します。ループバックアドレスは、実 IP アドレスの代わりに java.net.InetAddress.getLocalHost() によって返される場合があります。多くの場合、ループバックアドレスでは十分ではなく、Integration Server では実際のアドレスが必要になります。ホストの IP アドレスを DHCP サーバから動的に取得した場合、またはホストに複数のネットワークインタフェースカードが搭載されている場合に、このことが最もよく発生します。

ほとんどの場合は、C:\Windows\system32\drivers\etc file (Windows の場合)、あるいは etc/hosts ファイルまたは etc/nsswitch.conf ファイル (Linux および UNIX の場合) を変更すると IP アドレスを解決できます。これらのファイルを変更できない場合、またはこれらのファイルを変更しても問題が解決しない場合は、watt.net.localhost を設定します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.maxClientKeepaliveConns
特定のターゲットエンドポイント用に確保しておく、クライアントのキープアライブ接続のデフォルト数を設定します。デフォルトは 0 です。これは、Integration Server でターゲット

エンドポイントに対してクライアントのキープアライブ接続を保持しないことを示します。Integration Server は各要求の新しいソケットを作成します。

Software AG では、watt.net.maxClientKeepaliveConns を 0 に設定することをお勧めします。プロパティを 0 より大きい値に設定すると、特定のターゲットエンドポイントへの同時要求の頻度と数が多い状況で便利なことがあります。そのような状況ではない場合、アイドルソケットは失効状態で動作不能になり、次のような予期しない例外が発生します。

```
[ISC.0077.9998E] Exception --> org.apache.axis2.AxisFault: Broken pipe
```

watt.net.maxRedirects
I/O 例外を発行するまでに許容される HTTP リダイレクトの最大数を指定します。デフォルトは `5` です。

watt.net.maxRetries
失敗したソケット接続に対して Integration Server が実行できる再試行の最大回数を指定します。デフォルトは `1` です。値 `0` の場合、Integration Server は失敗したソケット接続を再試行しません。

watt.net.overrideSystemProxyselector
Java サービスがリモートサーバに接続を試みるときに、Integration Server のプロキシセレクタが、デフォルトの JVM システムプロキシセレクタを上書きするかどうかを指定します。このプロパティを「true」に設定すると、すべてのネットワーク接続で、Integration Server Administrator を使用して設定されたプロキシエイリアスが尊重されます。このプロパティを「false」に設定すると、デフォルトの JVM システムプロキシセレクタが使用されます。デフォルトは「`false`」です。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.primaryListener
これは内部プロパティです。変更しないでください。

watt.net.proxySkipList
Integration Server がプロキシサーバを使用できないドメイン名をカンマ区切りのリストで指定します。デフォルトは「`localhost`」です。

watt.net.proxy.fallbackToDirectConnection
要求されたプロトコルに対して指定されたすべてのプロキシサーバエイリアスを介した接続が失敗した場合、Integration Server が HTTP、HTTPS、FTP、SFTP および SOCKS 要求をターゲットサーバに直接ルーティングするかどうかを指定します。たとえば、要求が HTTP を使用している場合、Integration Server は要求を HTTP プロキシサーバエイリアス経由でルーティングします。このプロパティが「`false`」に設定され、プロキシエイリアスを介した宛先サーバへの接続が失敗した場合、Integration Server は例外を発行します。このプロパティが「`true`」に設定されている場合、Integration Server は要求で指定されている宛先サーバとの直接接続を試行します。デフォルトは「`true`」です。

> **注:**
> Integration Server にプロキシサーバエイリアスが定義されていない場合、watt.net.proxy.fallbackToDirectConnection の値は無視されます。プロキシサーバエイリア

スの詳細については、144 ページの「デフォルトのプロキシサーバエイリアスの指定」を参照してください。

watt.net.proxy.useNonDefaultProxies
送信要求でプロキシサーバエイリアスが指定されておらず、デフォルトのプロキシサーバエイリアスが指定されていない場合に、Integration Server がすべての有効なプロキシサーバエイリアスを使用して送信接続要求を行うかどうかを指定します。
watt.net.proxy.useNonDefaultProxies パラメータが「true」に設定されていると、送信要求でプロキシサーバエイリアスが指定されておらず、デフォルトのプロキシサーバエイリアスがない場合には、Integration Server は有効なプロキシサーバエイリアスを 1 つずつ使用して、要求の送信が成功するかすべてのプロキシサーバが試されるまで、送信要求を行います。すべてのプロキシサーバが試され、要求の送信の試行が失敗するか、またはプロキシエイリアスが指定されていない場合、Integration Server では、watt.net.proxy.fallbackToDirectConnection パラメータに指定された設定に基づいて、ターゲットサーバに直接接続するか、例外をスローします。watt.net.proxy.useNonDefaultProxies パラメータが「false」に設定され、デフォルトのプロキシサーバエイリアスが指定されていない場合は、Integration Server は直接接続を使用してリモートサーバに要求を送信します。Integration Server は、有効なプロキシサーバエイリアスを使用して送信要求を試行しません。デフォルトは「true」です。

Integration Server でプロキシサーバを使用する方法の詳細については、138 ページの「Integration Server によるプロキシサーバの使用方法」を参照してください。

watt.net.retries
タイムアウトになったサーバを再試行する回数を指定します。クライアントは watt.net.retries を上書きできます。デフォルトは 0 です。

watt.net.sftpSweepInterval
SFTP スイーパの実行間隔を分単位で指定します。SFTP スイーパはメモリ内の SFTP セッションを定期的に走査し、設定されたアイドル時間を過ぎたセッションを消去します。SFTP スイーパはデフォルトでは 10 分ごとに実行されます。

watt.net.socketpool.sweeperInterval
ソケットプールスイーパを実行する頻度を秒単位で指定します。ソケットプールスイーパは、すべての webMethods Enterprise Gateway 接続および HTTP クライアント接続に ping 要求を送信します。スイープ中に、無効な HTTP クライアント接続を削除します。デフォルトでは、スイーパは 60 秒ごとに実行されます。

注:
watt.net.socketpool.sweeperInterval の値は、watt.server.rg.internalregistration.timeout サーバ設定パラメータの値よりも小さくする必要があります。

注:
Enterprise Gateway で watt.server.rg.gateway.pinginterval が設定されている場合、Integration Server はこの設定を使用して、watt.net.socketpool.sweeperInterval の値を無視します。

watt.net.socketProvider
セキュアソケット通信用に com.wm.net.SocketProviderIf インタフェースを実装する Java クラスを示します。デフォルトは「com.wm.ext.iaik.IaikSecureSocket」です。

watt.net.ssl.client.cipherSuiteList

JSSE が使用されていないときに、送信 SSL 接続用の暗号スイートのリストを指定します。デフォルト値が `default` に設定されている場合、Integration Server は暗号スイートのデフォルトリストを使用します。デフォルト以外の暗号スイートを指定する場合、暗号スイート名をカンマ区切りのリストで入力します。watt.net.ssl.client.strongcipheronly プロパティを「`true`」に設定している場合、強力でない暗号スイートをリストに指定しても、それらの暗号スイートは無視され、警告メッセージがログに記録されます。

このパラメータの値はカンマ区切りリスト、デフォルト、またはファイルへの絶対パスに設定できます。上記の組み合わせを指定することはできません。このパラメータの値としてファイルを指定する方法の詳細については、488 ページの「SSL と使用する暗号スイートの指定」を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.client.handshake.maxVersion
Integration Server がクライアントとして機能して送信要求を行い、JSSE を使用していないときに、Integration Server でサポートされる SSL プロトコルの最大バージョンを指定します。たとえば、「`tls`」 (デフォルト) に設定した場合、Integration Server でサポートされる SSL プロトコルの最大バージョンは TLS 1.0 です。「`sslv3`」に設定した場合、Integration Server でサポートされる SSL プロトコルの最大バージョンは SSL 3.0 です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.client.handshake.minVersion
Integration Server がクライアントとして機能して送信要求を行い、JSSE を使用していないときに、Integration Server でサポートされる SSL プロトコルの最小バージョンを指定します。次のように設定されます。

- `sslv2` (デフォルト) SSL 2.0 の場合
- `tls` TLS 1.0 の場合
- `sslv3` SSL 3.0 の場合

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.client.hostnameverification
Integration Server が HTTPS クライアントとして機能しているときに、有効なホスト名がサーバ認証に見つかった場合にのみ Integration Server で送信 HTTPS 接続を許可するかどうかを指定します。

- 「`true`」に設定した場合、Integration Server でホスト名がサーバ認証にあるかどうかが検証されます。検証が失敗した場合、エラーのログが記録され接続は中断されます。
- 「`false`」に設定した場合、Integration Server ではホスト名の検証は行われません。これがデフォルトです。

- 「log」に設定した場合、Integration Server でホスト名の検証に失敗するとデバッグメッセージがサーバログに記録されますが、接続は許可されます。ホスト名の検証が成功した場合、ログは生成されません。

watt.net.ssl.client.strongcipheronly
Integration Server が、送信 HTTPS 接続に強力な暗号スイート (128 ビット以上のセッションキー) のみを使用するかどうかを指定します。「false」(デフォルト) を指定すると、Integration Server は別のサーバへの接続を開始するときに、強力な暗号スイートのネゴシエーションを行い、ネゴシエーションに失敗したときは弱い (64 ビット、56 ビットまたは 40 ビットの) 暗号スイートを使用します。「true」を指定すると、Integration Server は別のサーバへの接続を開始するとき、強力な暗号スイートのネゴシエーションを行い、ネゴシエーションに失敗したときは接続を切断します。弱い暗号スイートは使用しません。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.client.ftps.useJSSE
Integration Server から送信されるすべての FTPS 接続での JSSE の使用を制御します。すべての送信 FTPS 接続で JSSE を使用するには、このパラメータを「true」に設定します。このプロパティを「false」に設定すると、送信 FTPS 接続で JSSE を使用しないことを示します。デフォルトは「false」です。

注:
pub.client:ftp サービス、または pub.client.ftp:login サービスを実行する場合は、*useJSSE* 入力パラメータの値は watt.net.ssl.client.ftps.useJSSE サーバ設定パラメータの値を上書きします。

watt.net.ssl.client.useJSSE
Integration Server から送信されるすべての HTTPS 接続での JSSE の使用を制御します。すべての送信 HTTPS 接続で JSSE を使用するには、このパラメータを「true」に設定します。このプロパティを「false」に設定すると、送信 HTTPS 接続で JSSE を使用しないことを示します。デフォルトは「true」です。

注:
pub.client:http サービス、または pub.client:soapClient サービスを実行する場合は、*useJSSE* 入力パラメータの値は watt.net.ssl.client.useJSSE サーバ設定パラメータの値を上書きします。

watt.net.ssl.email.client.useJSSE
Integration Server のリソースに関するすべての電子メール通知に対して JSSE の使用を制御します。すべてのリソース通知で JSSE を使用するには、このパラメータを「true」に設定します。それ以外の場合は、「false」に設定します。デフォルト値は「true」です。

watt.net.ssl.email.client.smtp.useJSSE
Integration Server からのすべての送信 SMTP 接続に対して JSSE の使用を制御します。すべての送信 SMTP 接続で JSSE を使用するには、このパラメータを「true」に設定します。それ以外の場合は、「false」に設定します。デフォルト値は「false」です。

注:
pub.client:smtp サービスを実行する場合、サービスの *useJSSE* 入力パラメータの値がこのパラメータの代わりに使用されます。ただし、pub.client:smtp サービスの *useJSSE* 入力パラメータが空である場合、このパラメータが使用されます。

watt.net.ssl.randomAlgorithm
Integration Server で使用されるランダムアルゴリズム名を指定します。デフォルト値は「`FIPS186_2usingSHA1`」です。

注:
このパラメータは、Integration Server が HP-UX にインストールされているときだけ使用します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.cipherSuiteList
JSSE が使用されていないときに、受信 SSL 接続用の暗号スイートのリストを指定します。デフォルト値が `default` に設定されている場合、Integration Server は暗号スイートのデフォルトリストを使用します。デフォルト以外の暗号スイートを指定する場合、暗号スイート名をカンマ区切りのリストで入力します。watt.net.ssl.server.strongcipheronly プロパティを「`true`」に設定している場合、強力でない暗号スイートがリストにあっても、それらの暗号スイートは無視され、警告メッセージがログに記録されます。

このパラメータの値はカンマ区切りリスト、デフォルト、またはファイルへの絶対パスに設定できます。上記の組み合わせを指定することはできません。このパラメータの値としてファイルを指定する方法の詳細については、488 ページの「SSL と使用する暗号スイートの指定」を参照してください。

重要:
このパラメータの設定を変更した場合、変更を有効にするには、影響を受けるすべてのポートを再起動する必要があります。ポートを再起動するには、ポートを無効にしてから有効にします。ポートに関連付けられているパッケージを再ロードしたり、Integration Server を再起動しても、ポートが再起動されます。

watt.net.ssl.server.clientHandshakeTimeout
SSL ハンドシェーク中に Integration Server が応答を待機してからタイムアウトになるまでのミリ秒単位の数を指定します。Integration Server は受信/送信要求にこの値を使用します。デフォルトは 20000 ミリ秒です。

watt.net.ssl.server.handshake.maxVersion
送信要求を処理するサーバとして機能する Integration Server がサポートする SSL プロトコルの最大バージョンを指定します。次のように設定されます。

- `sslv3` SSL 3.0 の場合
- `tls` TLS 1.0 の場合

デフォルトは「`tls`」です。

watt.net.ssl.server.handshake.maxVersion の値は、SSL に Entrust ツールキットを使用するすべての HTTPS ポートに影響します。ポートの [**JSSE の使用**] パラメータが [**いいえ**] に設定されているときに、HTTPS ポートで Entrust を使用します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.handshake.minVersion
受信要求を処理するサーバとして機能する Integration Server がサポートする SSL プロトコルの最小バージョンを指定します。次のように設定されます。

- `sslv3` SSL 3.0 の場合
- `tls` TLS 1.0 の場合

デフォルトは「`tls`」です。

watt.net.ssl.server.handshake.minVersion の値は、SSL に Entrust ツールキットを使用するすべての HTTPS ポートに影響します。ポートの [**JSSE の使用**] パラメータが [**いいえ**] に設定されているときに、HTTPS ポートで Entrust を使用します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.sessionlog
Integration Server が受信接続に関する SSL セッション情報をログに記録するかどうかを指定します。このパラメータを「`true`」に指定した場合、Integration Server は Entrust および JSSE セキュリティプロバイダ用に用意する inboundSSLSessions.log ファイルに SSL セッションを記録します。Integration Server は *Integration Server_directory* \instances*instance_name*\logs ディレクトリに inboundSSLSessions.log ファイルを作成します。SSL セッション情報は JSON 形式でログに記録されます。このパラメータを「`false`」に設定すると、Integration Server は SSL セッションの情報をログに記録しません。デフォルト値は「`false`」です。

watt.net.ssl.server.sessionlog.maxFileSize
inboundSSLSessions.log ファイルの最大サイズを MB 単位で指定します。ファイルが最大サイズに達すると、Integration Server はファイルの名前を inboundSSLSessions_<DATE(YYYYMMDD)>_TIME(HHMMSS).log に変更し、新しい inboundSSLSessions.log ファイルを作成します。0 (ゼロ) よりも大きな整数を指定します。ゼロ以下の値を指定した場合、Integration Server はデフォルト値を使用します。デフォルト値は 10 です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.sessionlog.cacheLogEntries
Integration Server がキャッシュで SSL セッションログエントリを追跡するかどうかを指定します。このパラメータを「`true`」に設定した場合、Integration Server はキャッシュに既に存

在するエントリに関するSSLセッション情報をログに記録しません。これによって、ログファイルの重複エントリはなくなります。デフォルト値は「false」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.sessionlog.cachedLogEntries.expiryTime
Integration Server が期限切れの SSL セッションログエントリをチェックして、キャッシュから削除する頻度を秒単位で指定します。Integration Server がログエントリから削除したセッションをクライアントが再使用する場合、Integration Server はそのセッションのセッション情報を再度ログに記録します。0 (ゼロ) よりも大きな整数を指定します。ゼロ以下の値を指定した場合、Integration Server はデフォルト値を使用します。デフォルトは 300 秒です。

Integration Server は SSL セッションログエントリスイーパというスイーパタスクを使用して、期限切れセッションを削除します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.sessionlog.file
Integration Server が SSL セッション情報を書き込む先のファイルの絶対パスまたは相対パスを指定します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* *instances**instance_name* に相対的なパスです。パスには有効なディレクトリ名とファイル名を指定する必要があります。デフォルトは *Integration Server_directory* \instances*instance_name*\logs\inboundSSLSessions.log です。

注:
有効なパスを指定しなかった場合、Integration Server はデフォルトのパスとファイル名 *Integration Server_directory* \instances*instance_name*\logs\inboundSSLSessions.log を使用します。

ネットワーク上の場所を指すパスを指定した場合、その場所にアクセスできず、セッションの詳細をログに記録できないときは、その場所がアクセスできるようになるまで、Integration Server は SSL セッション情報をコンソールに出力します。さらに、Integration Server がセッションの詳細をログに記録するときに接続の問題が発生した場合、ログファイルには null 文字または無効な文字が記録されることがあります。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.sessionlog.prettyPrint
SSL セッションログを読みやすくするために、復帰 (CR) とインデントを使用して SSL セッションログエントリを書式設定するかどうかを指定します。このパラメータを「true」に設定した場合、Integration Server は復帰 (CR) およびインデントを使用して JSON 形式にすることにより読みやすくします。デフォルトは「false」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.sessionlog.includeTimestamp
Integration Server が SSL セッションログエントリにタイムスタンプを含めるかどうかを指定します。このパラメータを「true」に設定した場合、ログエントリはタイムスタンプで始まります。デフォルトは「false」です。

例:

このパラメータを「true」に設定した場合、ログエントリは次のようにタイムスタンプで始まります。`2019-07-03 10:37:24 IST {"provider":"JSSE","loggedInUser":"Administrator"…..}`。このパラメータを「false」に設定した場合、ログエントリには次のようにタイムスタンプは含まれません。`{"provider":"JSSE","loggedInUser":"Administrator"…..}`。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.net.ssl.server.strongcipheronly
Integration Server が、受信 HTTPS 接続に強力な暗号スイート (128 ビット以上のセッションキー) のみを使用するかどうかを指定します。「`false`」 (デフォルト) を指定すると、クライアントが Integration Server に接続するときに、サーバは強力な暗号スイートのネゴシエーションを行い、ネゴシエーションに失敗したときは弱い (64 ビット、56 ビットまたは 40 ビットの) 暗号スイートを使用します。「`true`」を指定すると、クライアントが Integration Server に接続するとき、サーバは強力な暗号スイートのネゴシエーションを行い、ネゴシエーションに失敗したときは接続を切断します。弱い暗号スイートは使用しません。

> **重要:**
> このパラメータの設定を変更した場合、変更を有効にするには、影響を受けるすべてのポートを再起動する必要があります。ポートを再起動するには、ポートを無効にしてから有効にします。ポートに関連付けられているパッケージを再ロードしたり、Integration Server を再起動しても、ポートが再起動されます。

watt.net.timeout
サーバが HTTP 要求の実行をタイムアウトになるまで待機する秒数を指定します。ターゲットサーバからの応答を無期限に待機するように Integration Server を設定するには、このパラメータを「`0`」に設定します。デフォルトは `300` (5 分) です。

> **重要:**
> watt.net.timeout を「`0`」に設定した場合に、ターゲットサーバが要求に応答しないと、要求を行った Integration Server はスレッドプールの枯渇により新しい要求を処理できません。

watt.net.useCookies
Integration Server で Web サーバとの通信時に Cookie を受け入れるか、拒否するかを指定します。Cookie を受け入れる場合は「true」に設定し、拒否する場合は「false」 (または NULL) に設定します。デフォルトは「`true`」です。

watt.net.userAgent
サーバが Web サーバから Web ドキュメントを要求するときに HTTP User Agent 要求ヘッダーで使用する値を指定します。デフォルトは「`Mozilla/4.0 [en] (WinNT; I)`」です。

watt.security.

watt.security.cert.wmChainVerifier.enforceExtensionsChecks
サーバでの認証検証時に Integration Server で認証拡張子 (存在する場合) の妥当性を検査するか (`true`)、しないか (`false`) を指定します。デフォルトは「`false`」です。

watt.security.cert.wmChainVerifier.trustByDefault
[セキュリティ] > [認証] ページの [**トラストストアエイリアス**] フィールドにトラストストアエイリアスが設定されていない場合、Integration Server が次の認証および署名を信頼するかどうかを指定します。

- (サーバの送信要求に応じて) ピアサーバが提示する認証
- S/MIME 署名

上記の場合に Integration Server が認証および S/MIME 署名を信用する (`true`)、または信用しない (`false`) ことを指定します。デフォルトは「`true`」です。Software AG では、セキュリティを向上させるために、このパラメータを「`false`」に設定し、**[セキュリティ] > [認証]** ページの [**トラストストアエイリアス**] フィールドにトラストストアエイリアスを設定することをお勧めします。

watt.security.decrypt.keyAlias
Integration Server で復号化に使用されるデフォルトの秘密鍵のエイリアスを指定します。このパラメータのデフォルト値はありません。

watt.security.decrypt.keyAlias パラメータは、[**セキュリティ**] > [認証] ページの [**複合キー**] 下の [**キーエイリアス**] フィールドに設定された値を表示します。

watt.security.decrypt.keyStoreAlias
Integration Server で復号化に使用されるデフォルトの秘密鍵を格納するキーストアのエイリアスを指定します。デフォルト値はありません。

watt.security.decrypt.keyStoreAlias パラメータは、**[セキュリティ] > [認証]** ページの [**複合キー**] 下の [**キーストアエイリアス**] フィールドに設定された値を表示します。

watt.security.fips.mode
サーバが FIPS (Federal Information Processing Standards : 米連邦情報処理規格) をサポートするかどうかを指定します。デフォルトは「`false`」です。このパラメータが「`true`」に設定されている場合、サーバはサーバ起動の一環として FIPS を初期化します。FIPS の初期化に失敗した場合は、エラーのログが server.log に記録され、サーバがシャットダウンします。

watt.security.kerberos.client.useSPNEGO
Integration Server が、ケルベロス認証を使用するすべての送信要求に対して SPNEGO ベースのケルベロスチケットを生成するかどうかを指定します。「true」に設定すると、ケルベロス認証を使用するすべての送信要求に対して Integration Server が SPNEGO OID (オブジェクト識別子) を使用して SPNEGO トークンを生成します。「false」に設定すると、Integration Server は JGSS ケルベロス OID を使用して Java ケルベロスチケットを生成します。このプロパティのデフォルト値は「false」です。

watt.security.KeystoreAndTruststore.defaultAliasCreated
これは内部パラメータです。変更しないでください。

watt.security.keyStore.supportedTypes

Integration Server でサポートされているキーストアタイプを指定します。サポートされている各タイプをカンマで区切ります。デフォルト値は「JKS」および「PKCS12」です。

Integration Server の追加のキーストアタイプを指定するには、次の処理が必要です。

1. このプロパティの値のリストに新しいキーストアタイプを追加します。
2. 次のいずれかの方法を使用して、キーストアプロバイダを追加します。
   a. Integration Server Administrator の [セキュリティプロバイダの追加] リンクを使用します。
   b. JVM の「java.security」ファイルを変更します (PKCS11 タイプのキーストアにはこの方法を使用する必要があります)。

> **注:**
> Software AG では、追加のキーストアタイプの動作は保証されず、サポートされていません。

watt.security.max.contentLength

HTTP 要求に対してペイロードの最大サイズ制限を指定します。watt.security.max.contentLength が 0 より大きく、Integration Server が watt.security.max.contentLength より大きいコンテンツの長さヘッダーの HTTP 要求を受信した場合、Integration Server は「413 ペイロードが大きすぎます」というエラーと共に要求を拒否し、接続を閉じます。

watt.security.max.contentLength が 0 より大きく、Integration Server が要求ヘッダーに Content-Length がない要求を受信する場合、Integration Server は「411 長さが必要です」というエラーと共に要求を拒否し、接続を閉じます。このパラメータの値を 0 より大きく設定し、Content-Length の最大値を適用します。watt.security.max.contentLength のデフォルトの値は 0 です。これは、Integration Server が Content-Length の最大値を適用しないことを意味します。

> **注:**
> watt.security.max* サーバ設定パラメータを設定する際は注意してください。設定されたパラメータの値が低すぎる場合、Integration Server は正当な要求を拒否します。パラメータを低く設定しすぎると、Integration Server Administrator からロックアウトしてしまう可能性があります。このような場合は、コマンドライン HTTP クライアントと共に wm.server.admin:setSettings サービスを呼び出すことによって、設定パラメータの値の変更を試みます。これらのパラメータの値を変更できない場合は、Integration Server を強制終了し、手動で *Integration Server_directory* /instances/*instanceName*/config/server.cnf ファイルを編集し、Integration Server を再起動する必要があります。

watt.security.max.headerLength

HTTP 要求に対してヘッダーの最大長を指定します。watt.security.max.headerLength が 0 より大きく、Integration Server が受信した要求に watt.security.max.headerLength より長いヘッダーがある場合、Integration Server は「431 要求ヘッダーフィールドが長すぎます」というエラーと共に要求を拒否し、接続を閉じます。ヘッダーの長さを算定する場合、Integration Server は要求ヘッダーのすべての要求ヘッダーフィールドのキーと値、およびすべての空白文字を考慮します。このパラメータの値を 0 より大きく設定し、要求ヘッダーの最大長を適用します。watt.security.max.headerLength のデフォルトの値は 0 です。これは、Integration Server がヘッダーの最大長を適用しないことを意味します。

> **注:**

watt.security.max* サーバ設定パラメータを設定する際は注意してください。設定されたパラメータの値が低すぎる場合、Integration Server は正当な要求を拒否します。パラメータを低く設定しすぎると、Integration Server Administrator からロックアウトしてしまう可能性があります。このような場合は、コマンドライン HTTP クライアントと共に wm.server.admin:setSettings サービスを呼び出すことによって、設定パラメータの値の変更を試みます。これらのパラメータの値を変更できない場合は、Integration Server を強制終了し、手動で *Integration Server_directory* /instances/*instanceName*/config/server.cnf ファイルを編集し、Integration Server を再起動する必要があります。

watt.security.max.urlLength

HTTP 要求に対して URL の最大長を指定します。watt.security.max.urlLength が 0 より大きく、Integration Server が watt.security.max.urlLength より長い URL の要求を受信する場合、Integration Server は「414 URL が長すぎます」というエラーと共に要求を拒否し、接続を閉じます。URL の長さを算出する場合、Integration Server は URL のパスおよびクエリーの部分を考慮します。このパラメータの値を 0 より大きく設定し、URL の最大長を適用します。watt.security.max.urlLength のデフォルトの値は 0 です。これは、Integration Server が URL の最大長を適用しないことを意味します。

注:

watt.security.max* サーバ設定パラメータを設定する際は注意してください。設定されたパラメータの値が低すぎる場合、Integration Server は正当な要求を拒否します。パラメータを低く設定しすぎると、Integration Server Administrator からロックアウトしてしまう可能性があります。このような場合は、コマンドライン HTTP クライアントと共に wm.server.admin:setSettings サービスを呼び出すことによって、設定パラメータの値の変更を試みます。これらのパラメータの値を変更できない場合は、Integration Server を強制終了し、手動で *Integration Server_directory* /instances/*instanceName*/config/server.cnf ファイルを編集し、Integration Server を再起動する必要があります。

watt.security.min.bytesPerSecond

Integration Server にデータが到着するときの最小レートを指定します。watt.security.min.bytesPerSecond の値よりも遅い速度でデータが到着した場合、Integration Server は、要求を「408 Request Timeout」エラーにより拒否し、接続を閉じます。レートは各要求ごとに計算されます。Integration Server により要求が受信されるたびに、Integration Server がソケットからバイトを読み取るレートが監視されます。Integration Server がソケットの InputStream の最後を読むと、Integration Server は要求の監視を停止します。

データの到着時の最小レートを適用するには、このパラメータの値を 0 より大きく設定します。value for watt.security.min.bytesPerSecond のデフォルトの値は 0 です。これは、Integration Server にデータ受信時の最小レートを適用しないことを意味します。

Integration Server は、Integration Server が実際にバイトを読んでいる要求に対してのみ watt.security.min.bytesPerSecond 値を適用できます。場合によっては、バイトの読み取りが顧客のアプリケーションやサードパーティのライブラリによって制御されていることがあります。このような場合は、Integration Server は受信バイトを読み取っておらず、受信時のレートを監視できません。次のようなケースでは、Integration Server は watt.security.min.bytesPerSecond の値を適用しません。

- 要求ペイロードが JSON および watt.server.http.jsonFormat がストリームの場合
- 要求ペイロードが XML および watt.server.http.xmlFormat がストリームの場合
- 要求に Content-type ヘッダーが含まれていない場合

- 拡張 XML パーサが使用されている場合
- pub.xml NodeIterator サービスが使用されている場合
- SOAP 要求

アプリケーションがこれらの機能を使用する場合、watt.security.min.bytesPerSecond をより低い値に設定する必要があることがあります。

watt.security.min.bytesPerSecond 設定パラメータを設定する際は注意が必要です。パラメータの値を高く設定しすぎると、Integration Server が遅いネットワークを介して送られた正当な要求を拒否することがあります。また、パラメータを高く設定しすぎると、自分自身を Integration Server Administrator からロックアウトしてしまう可能性があります。このような場合は、コマンドライン HTTP クライアントを使って wm.server.admin:setSettings サービスを呼び出すことによって、設定パラメータの値の変更を試みます。このパラメータの値を変更できない場合は、Integration Server を強制終了し、手動で *Integration Server_directory* /instances/*instanceName*/config/server.cnf ファイルを編集し、Integration Server を再起動する必要があります。

watt.security.ope.AllowInternalPasswordAccess

OPE (Outbound Password Encryption : 送信パスワード暗号化) をサポートするフローサービスの組み込みサービスに Integration Server の内部パスワードへのアクセスを許可するかどうかを指定します。このパラメータを「`true`」に設定した場合、OPE サービスは内部パスワードにアクセスできます。このパラメータを「`false`」に設定した場合、OPE サービスは内部パスワードへのアクセスを許されません。デフォルトでは、このパラメータは「`false`」に設定されます。

内部パスワードとは、Integration Server 自体がセキュアリソース (たとえば、リモート Integration Server、JDBC 接続プール、LDAP サーバなど) にアクセスするために使用するパスワードです。内部パスワードは Integration Server Administrator を使用して管理され、送信パスワードストアに保存されます。フローサービスもまた、送信パスワードストアにパスワードを保存できます。ただし、デフォルトでは、フローサービスによって保存されたパスワードは、「内部」ではなくパブリックと見なされます。このような区別により、Integration Server の内部パスワードを保護しながら、フローサービスがパスワードの保存と取得のための安全なメカニズムとして送信パスワードストアを使用することが可能になります。

watt.security.ope.AllowInternalPasswordAccess を「`true`」に設定すれば、フローサービスによる内部パスワードへのアクセスを許可できます (保存、取得、変更など)。ただし、フローサービスで内部パスワードを操作する必要性が明確に認識されている場合以外、この設定は使用しないでください。それ以外の場合は、watt.security.ope.AllowInternalPasswordAccess を「`false`」に設定して、内部パスワードへのアクセスを拒否することをお勧めします。

watt.security.openid.logExceptions

Integration Server がエラーログに OpenID エラーを書き込むかどうかを指定します。OpenID 認証プロセスでエラーが発生すると、Integration Server は HTTP 応答の本文で *error* および *error_description* 変数を返します。watt.security.openid.logExceptions をデフォルトの「true」に設定すると、Integration Server はリダイレクトエンドポイントサービスから例外をスローし、エラーログにはエラーが書き込まれます。Integration Server が OpenID エラーをエラーログに書き込むことを停止するには、watt.security.openid.logExceptions を「false」に設定します。

watt.security.pub.getFile.checkReadAllowed

要求されたファイルをファイルシステムから取得できるかどうかを確認するために、pub.file:getFile サービスで fileAccessControl.cnf ファイルの allowedReadPaths プロパティをチェックするかどうかを指定します。allowedReadPaths プロパティには、pub.file フォルダのサービスが読み取り権限を持つディレクトリのリストが格納されます。

watt.security.pub.getFile.checkReadAllowed を「`true`」に設定すると、pub.file:getFile サービスでは、fileAccessControl.cnf ファイルの allowedReadPaths プロパティをチェックします。要求されたファイルまたはファイルディレクトリがリストされている場合、pub.file:getFile サービスはその内容をパイプラインに返します。

watt.security.pub.getFile.checkReadAllowed を「`true`」に設定した場合に、そのファイルまたはファイルディレクトリが allowedReadPaths プロパティにリストされていないと、pub.file:getFile サービスでは例外をスローします。

watt.security.pub.getFile.checkReadAllowed を「`false`」に設定すると、pub.file:getFile サービスでは、fileAccessControl.cnf ファイルをチェックすることなく、指定されたファイルをローカルファイルシステムから取得します。

このプロパティの変更は即座に反映されます。ただし、fileAccessControl.cnf ファイルを変更した場合、変更内容を有効にするには WmPublic パッケージを再ロードする必要があります。

pub.file:getFile サービスの詳細および fileAccessControl.cnf ファイルの設定については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.security.session.forceReauthOnExpiration
Integration Server が期限切れまたは無効なセッションが含まれる要求を受け入れるか拒否するかを指定します。「true」に設定すると、Integration Server は、要求に有効なユーザクレデンシャルが含まれる場合でも、期限切れまたは無効なセッションを指定している Cookie を含む要求を拒否します。拒否応答は、ブラウザにセッション ID をクリアさせ、ユーザにクレデンシャルの入力を促します。「false」に設定すると、Integration Server は Cookie のクレデンシャルを使用して新しいセッションを作成します。デフォルト値は true です。値を「true」にすると、動作のセキュリティが高くなります。

watt.security.sign.keyAlias
Integration Server でドキュメントのデジタル署名に使用されるデフォルトの秘密鍵のエイリアスを指定します。このパラメータのデフォルト値はありません。

watt.security.sign.keyAlias パラメータは、**[セキュリティ] > [認証]** ページの [**署名キー**] 下の [**キーエイリアス**] フィールドに設定された値を表示します。

watt.security.sign.keyStoreAlias
Integration Server でドキュメントのデジタル署名に使用されるデフォルトの秘密鍵を格納するキーストアのエイリアスを指定します。このパラメータのデフォルト値はありません。

watt.security.sign.keyStoreAlias パラメータは、**[セキュリティ] > [認証]** ページの [**署名キー**] 下の [**キーストアエイリアス**] フィールドに設定された値を表示します。

watt.security.ssl.cacheClientSessions
Integration Server が同じクライアントへの接続に対して以前の SSL セッション情報 (たとえばクライアント認証) を再利用するかどうかを制御します。このプロパティを「`true`」に設定すると、Integration Server は SSL セッションの情報をキャッシュして再利用します。たとえば、同じクライアントから HTTPS 要求が繰り返される場合は、このプロパティを「`true`」に設定します。「`false`」 (デフォルト) に設定すると、Integration Server はセッションをキャッシュせず、SSL ハンドシェークごとに新しいセッションを作成します。

注:
このプロパティを「false」に設定すると、パフォーマンスが低下することがあります。

watt.security.ssl.cachedClientSessions.sweeperInterval
Integration Server で期限切れの SSL クライアントセッションをチェックして、セッションキャッシュから削除する頻度をミリ秒単位で指定します。デフォルト値は 600000 ミリ秒 (10 分) です。

watt.security.ssl.client.ignoreEmptyAuthoritiesList
リモート SSL サーバが信用のある認証局の空のリストを返した後、クライアントとして動作する Integration Server が認証チェーンを送信するかどうかを指定します。「true」に設定すると、Integration Server は信用のある認証局の空のリストを無視し、チェーンを送信します。「false」に設定すると、Integration Server では、認証チェーンを送信する前に、それ自体が信用のあることを証明する、信用のある認証局リストの提示が必要になります。デフォルトは「false」です。

watt.security.ssl.ignoreExpiredChains
Integration Server が、Web サーバやその他の Integration Server などのインターネットリソースから受信した認証チェーン内の期限切れの CA 認証を無視するかどうかを指定します。Integration Server で期限切れの CA 認証を無視し、認証の期限が切れている場合に SSL 接続を許可するには、watt.security.ssl.ignoreExpiredChains を「true」に設定します。この方法では、認証が期限切れの場合に接続を拒否するよりも安全性が低くなります。デフォルトは「false」です。この設定の詳細については、を参照してください。

watt.security.ssl.keyAlias
Integration Server のデフォルトの SSL 秘密鍵のエイリアスを指定します。このパラメータのデフォルト値はありません。

watt.security.ssl.keyAlias パラメータは、**[セキュリティ] > [認証]** ページの [**SSL キー**] 下の [**キーエイリアス**] フィールドに設定された値を表示します。

watt.security.ssl.keypurposeverification
Integration Server が HTTPS クライアントとして機能している場合、有効な Extended Key Purpose フィールドがサーバ認証に存在する場合にのみ送信 HTTPS 接続を許可するかどうかを指定します。検証に成功するためには、Key Purpose フィールドの内容 (id-kp-serverAuth) は、IETF 指定の形式 (TLS WWW server authentication) に従っている必要があります。この形式の詳細については、URL http://www.ietf.org/rfc/rfc3280.txt のドキュメントの「Extended Key Usage」セクションを参照してください。

この watt プロパティに指定できる値は、「true」、「false」および「log」です。

- 「true」に設定した場合、サーバ認証に Key Purpose フィールドが存在するかどうかが検証されます。Key Purpose フィールドの検証が失敗した場合、エラーのログが記録され接続は中断されます。検証が成功した場合、エラーのログは記録されません。
- 「false」に設定した場合、Key Purpose フィールドの検証は行われません。デフォルトは「false」です。
- 「log」に設定した場合、Key Purpose フィールドの検証に失敗するとデバッグメッセージがサーバログに記録されます。

watt.security.ssl.keyStoreAlias

Integration Server のデフォルトの SSL キーストアのエイリアスを指定します。このパラメータのデフォルト値はありません。

watt.security.ssl.keyStoreAlias パラメータは、**[セキュリティ] > [認証]** ページの [**SSL キー**] 下の [**キーストアエイリアス**] フィールドに設定された値を表示します。

watt.security.ssl.resumeClientSessions
クライアントとして機能する Integration Server が同じサーバへの接続に前の SSL セッション情報を再使用するかどうかを制御します。

- watt.security.ssl.resumeClientSessions が「`true`」に設定された場合、さらに
  - Integration Server が HTTPS クライアントとして機能する場合、Integration Server は SSL セッションをキャッシュし、同じサーバへの将来の要求に対して SSL セッション情報を再使用します。
  - Integration Server が FTPS クライアントとして機能する場合、Integration Server はコマンドとデータチャネルに同じ SSL セッションを使用します。
- watt.security.ssl.resumeClientSessions が「`false`」に設定された場合、さらに
  - Integration Server が HTTPS クライアントとして機能する場合、Integration Server は SSL セッションをキャッシュせず、すべての SSL ハンドシェークに新しい SSL セッションを作成します。
  - Integration Server が FTPS クライアントとして機能する場合、SSL セッションをキャッシュせず、コマンドとデータチャネルに新しい SSL セッションを作成します。デフォルト値は「false」です。

watt.security.trustStore.supportedTypes
Integration Server でサポートされているトラストストアタイプを指定します。サポートされている各タイプをカンマで区切ります。デフォルト値は「JKS」です。

Integration Server の追加のトラストストアタイプを指定するには、次の処理が必要です。

1. このプロパティの値のリストに新しいトラストストアタイプを追加します。
2. 次のいずれかの方法を使用して、トラストストアプロバイダを追加します。
   a. Integration Server Administrator の [セキュリティプロバイダの追加] リンクを使用します。
   b. JVM の「java.security」ファイルを変更します。

   注:
   Software AG では、追加のトラストストアタイプの動作は保証されず、サポートされていません。

watt.security.trustStoreAlias
Integration Server でデフォルトトラストストアとして使用されるトラストストアのエイリアスを指定します。

Integration Server がサーバとして機能している場合、トラストストアが提供されていなければ、デフォルトトラストストアを使用して信用関係が検証されます。たとえば、Integration Server では、HTTPS/FTPS ポートに対してトラストストアが提供されていない場合にデフォルトトラストストアが使用されます。また、プロバイダ Web サービスエンドポイントエイリ

アスにトラストストアが存在しない場合、Web サービスセキュリティにデフォルトトラストストアが使用されます。

Integration Server がクライアントとして機能している場合、Integration Server のほとんどのコンポーネント (HTTPS、FTPS、リモートサーバエイリアスなど) では常にデフォルトのトラストストアを使用して、サーバとの信用関係が検証されます。ただし、Web サービスセキュリティに対して Integration Server でデフォルトトラストストアが使用されるのは、ユーザがコンシューマエンドポイントエイリアスでトラストストアを提供していない場合だけです。

watt.security.trustStoreAlias パラメータは、**[セキュリティ] > [認証]** ページの [**トラストストア**] 下の [**トラストストアエイリアス**] フィールドに設定された値を表示します。

watt.server.

watt.server
これは内部パラメータです。変更しないでください。

watt.server.acl.groupScanInterval
Integration Server の起動時にデータベースが使用できない場合に、My webMethods Server で Integration Server データベースが使用可能かどうかをチェックする頻度をミリ秒単位で指定します。データベースが使用できない場合、セントラルユーザ管理で管理されるユーザに関するグループ情報は参照できません。そのため、これらのグループ内のユーザは、関連する ACL によって付与されるアクセス権を持ちません。My webMethods Server データベースが使用できない状態の間、[**セキュリティ**] > [ACL] ページには、影響を受けるユーザがサフィックス「`@--@`」付きで表示されます (「`FinanceGroup@--@`」など)。My webMethods Server が使用可能になると、グループはアクセス権を取り戻し、Integration Server によってサフィックス「`@--@`」が削除されます。デフォルト設定は 10,000 ミリ秒 (10 秒) です。

watt.server.allowDirective
指定されたディレクティブの使用を指定のポートに制限します。ディレクティブの詳細については、532 ページの「ディレクティブの使用の制御」を参照してください。このプロパティの構文は次のとおりです。

port-string は、「5555,6666」など、ポート番号のカンマ区切りリストです。

すべてのポートにデフォルトのディレクティブの使用を許可するが、以下に示すように特定のポートについては他のディレクティブを使用するように指定するとします。

invoke ディレクティブの使用をポート 5555 および 7777 に制限

web ディレクティブの使用をポート 6666 および 7777 に制限

SOAP ディレクティブの使用をポート 7777 に制限

上記の動作を実現するには、次のように指定します。

```
watt.server.allowDirective=invoke,5555,7777,web,6666,7777,soap,7777
```

watt.server.apiportal.url
API Portal への接続を確立し、REST API 記述子をパブリッシュするための URL を指定します。このパラメータは、以下の項目から API Portal URL を取得します。

- API Portal 接続のホスト名とポート番号
- REST API 記述子のパブリッシュ対象のテナント

- REST API 記述子の仕様

watt.server.audit.dbEncoding
監査ログデータベースによって使用される文字セットを指定します。デフォルトは「UTF-8」です。このプロパティの値は、IANA Character Sets 仕様書に定義されているとおりの、IANA (Internet Assigned Numbers Authority) が割り当てた標準の文字セット名でなければなりません。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.disableInternalServerConnectionSuccessLogging
セキュリティログに「内部サーバが Enterprise Gateway に正常に認証されました。」というエントリを含めない場合は、「true」に設定します。デフォルト値は「false」です。

注:
このプロパティが「true」の場合、IS Administrator の [**ログ**] > [ログ設定] > [セキュリティロガーの編集] ページの [**監査データを生成するタイミング**] が「成功時」または「成功時または失敗時」に設定されていても適用されます。

watt.server.audit.displayLogs.convertTime
Integration Server Administrator でサービス、エラー、セッション、保証付きデリバーおよびセキュリティのログを表示するときの日付の表示形式を指定します。このプロパティを「false」に設定した場合、タイムスタンプは GMT/UTC として表示されます。このプロパティを「true」(デフォルト) に設定した場合、タイムスタンプは Integration Server のローカルタイムゾーンで表示され、watt.server.dateStampFmt で指定した形式になります。

watt.server.audit.failFastLoggers
ISCoreAudit 機能エイリアスでフェイルファストモードが有効になっている場合に、フェイルファストモードを有効にできるロガーを指定します。

- watt.server.audit.failFastLoggers が空の場合は、データベースの宛先を使用するすべての同期監査ロガーがフェイルファスト機能を持ちます。
- watt.server.audit.failFastLoggers が空でない場合は、watt.server.audit.failFastLoggers に列挙されたデータベース宛先を持つ同期監査ロガーだけがフェイルファスト機能を持ちます。

注:
ISCoreAudit 機能エイリアスで [**フェイルファストモードが有効**] オプションを [**はい**] に設定すると、そのエイリアスでフェイルファストモードが有効になります。

フェイルファストモードするための特定の監査ロガーを指定する場合は、watt.server.audit.failFastLoggers を使用してフェイルファストモードを有効にするロガーのカンマ区切りリストを指定します。サービスロガーとセッションロガーでフェイルファストモードを有効にする場合は、プロパティを次のように設定します。

```
watt.server.audit.failFastLoggers=Service Logger,Session Logger
```

「Logger」という単語をロガー名の一部に指定する必要があります。プロパティでは大文字と小文字が区別されます。Integration Server Administrator の [**ログ**] > [ログ設定] ページに

表示されるものと 同じロガー名を指定する 必要があります。ロガー名を区切るカンマの前後にスペースを入れないでください。

フェイルファストモードを有効にできるのは、同期ロガーだけです。Integration Server は、watt.server.audit.failFastLoggers プロパティで指定された非同期ロガーは無視します。

watt.server.audit.failFastLoggers のデフォルトは空です。つまり、ISCoreAudit 機能で [**フェイルファストモードが有効**] オプションに [**はい**] を指定すると、データベースの宛先を使用するすべての同期監査ロガーはフェイルファスト機能を持ちます。

watt.server.audit.file.fieldDelimiter
ファイルベースの監査ログのログレコード内でフィールドの区切りに使用する文字シーケンスを指定します。有効な区切り文字は、US ASCII 文字セットの表示可能 (印刷可能) 文字 (0021～007E) を使用した文字シーケンスです。区切り文字シーケンスを指定する場合、ログエントリに含まれる可能性が高い文字シーケンスは使用しないでください。ファイルベースの監査ログで区切り文字を使用するには、watt.server.audit.file.fieldDelimiter および watt.server.audit.file.recordDelimiter パラメータの有効な値を指定します。両方のパラメータに有効な値を指定しなかった場合、Integration Server は固定長形式を使用してログエントリを書き込みます。

このパラメータのデフォルト値はありません。デフォルトで、Integration Server はファイルベースの監査ログに固定長のフィールドエントリを使用します。

ファイルベースの監査ログに対して固定長のフィールドエントリに戻すには、watt.server.audit.file.fieldDelimiter および watt.server.audit.file.recordDelimiter パラメータを削除し、Integration Server を再起動します。

固定長形式と文字区切り形式間での切り替え後、Integration Server Administrator は前の形式のエントリを表示できません。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.file.recordDelimiter
ファイルベースの監査ログでのレコードの区切りに使用する文字シーケンスを指定します。有効な区切り文字は、US ASCII 文字セットの表示可能 (印刷) 文字 (0021～007E) を使用した文字シーケンスです。区切り文字シーケンスを指定する場合、ログエントリに含まれる可能性が高い文字シーケンスは使用しないでください。ファイルベースの監査ログで区切り文字を使用するには、watt.server.audit.file.fieldDelimiter および watt.server.audit.file.recordDelimiter パラメータの有効な値を指定します。両方のパラメータに有効な値を指定しなかった場合、Integration Server は固定長形式を使用してログエントリを書き込みます。

このパラメータのデフォルト値はありません。デフォルトで、Integration Server はファイルベースの監査ログに固定長のフィールドエントリを使用します。

ファイルベースの監査ログに対して固定長のフィールドエントリに戻すには、watt.server.audit.file.fieldDelimiter および watt.server.audit.file.recordDelimiter パラメータを削除し、Integration Server を再起動します。

固定長形式と文字区切り形式間での切り替え後、Integration Server Administrator は前の形式のエントリを表示できません。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.journal.truncated
監査ログ内で切り捨てられたフィールドの完全な値をサーバログに書き込むときのログレベルを指定します。Integration Server が監査ログを書き込むときに、フィールドがデータベースの列に入れるには大きすぎる場合、Integration Server は監査ログでは値を切り捨て、フィールドの完全な値をサーバログに書き込みます。サーバログ機能である「**0095 Audit Log Manager**」が watt.server.audit.journal.truncated と同じレベル、または更に冗長なレベルに設定されている場合に適用されます。例えば、「**0095 Audit Log Manager**」のログレベルが「Info」に設定されていて、watt.server.audit.journal.truncated が「Debug」に設定されている場合、切り捨てられたフィールドの完全値はサーバログに現れません。「**0095 Audit Log Manager**」のログレベルが「Trace」に設定されていて、watt.server.audit.journal.truncated が「Debug」に設定されている場合、切り捨てられたフィールドの完全値はサーバログに現れます。

有効な値は、Critical、Error、Warn、Info、Debug、Trace および Off です。デフォルト値は「Info」です。

watt.server.audit.logDir
監査ログファイルを格納するディレクトリを指定します。完全パス名または Integration Server に対する相対パスを指定できます。このディレクトリはサーバ上に存在する必要があります。デフォルト値は `logs` であり、これは *Integration Server_directory* \instances*instance_name*\logs ディレクトリを示します。無効な値を指定すると、Integration Server では、デフォルト値を使用して、起動時にサーバログにエラーを書き込みます。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.logDir は、データベースに書き込む監査ロガーには影響を及ぼさないことに注意してください。

watt.server.audit.logFilesToKeep
現在のログファイルを含む、監査ロガー用の監査ログファイルの数を指定します。Integration Server はファイルに書き込む監査ロガーをファイルシステムに保存し続けます。Integration Server が監査ロガーのログファイル数の上限に達すると、Integration Server は監査ログを循環するごとに、アーカイブ済みの最も古い監査ログファイルを削除します。
watt.server.audit.log.filesToKeep を 1 に設定すると、Integration Serverは現在の監査ログファイルは保持しますが、各ファイルシステムベースの監査ロガー用の古い監査ログファイルは保持しません。つまり、Integration Server がロガー用の監査ログを循環する場合、Integration Server は古い監査ログ用のアーカイブファイルを作成しません。
watt.server.audit.logFilesToKeep を 0 または 1 未満の値に設定した場合、Integration Server は無制限の数の監査ログファイルを保持します。

watt.server.log.filesToKeep のデフォルト値は 0 です。これは、ファイルに書き込む監査ロガーに対して Integration Server が保持し続ける監査ログファイル数に制限がないことを意味します。

watt.server.audit.logFilesToKeep パラメータは、ファイルへ書き込むように設定された監査ロガーにのみ影響します。このパラメータは、データベースへの書き込みが設定された監査ロガーには影響せず、FailedAuditLog にも影響しません。

Integration Server がファイルベースの監査ログ用に保持するログの数を減らしてから Integration Serverを再起動した場合、Integration Server が監査ログに書き込むまで、既存の監査ログは削除されません。たとえば、エラーロガーがファイルに書き込まれていて、保持

されるログファイルの数を 10 から 6 に減らした場合、Integration Server の起動直後には古い 4 つのエラー監査ログファイルは削除されません。エラーロガーがエラー監査ログに書き込んだ後、Integration Server は 4 つのエラー監査ログを古い順に削除します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.logRotateSize
Integration Server がファイルに書き込むロガーの監査ログをロールオーバするファイルサイズを指定します。このプロパティは、*N*[KB|MB|GB] に設定します。*N* は、有効な任意の整数です。Integration Server が監査ログを循環する最小サイズは 33 KB です。単位として KB を使用する場合、*N* は 33 以上の値に設定する必要があります。単位を指定しない場合、Integration Server では *N* 値をバイトとして扱います。この場合、*N* は 32768 以上に設定しないと有効になりません。整数と測定単位の間にスペースを入れないでください。

このパラメータのデフォルト値はありません。デフォルトでは、パラメータの値はありません。watt.server.audit.logRotateSize の値が指定されない場合は、Integration Server は監査ログを午前零時にのみ循環します。

無効な値が指定された場合、Integration Server はパラメータの値が指定されていないかのように処理を続行します。Integration Server Administrator の [**拡張設定**] ページを使用して値を設定した場合、妥当性検査によって無効な値は保存されません。

watt.server.audit.logRotateSize パラメータは、ファイルへの書き込みが設定された監査ロガーにのみ影響します。パラメータは、データベースへの書き込みが設定された監査ロガーには影響しません。

監査ログの詳細については、『*webMethods Audit Logging Guide*』を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.schemaName
Integration Server が監査ログデータベースのメタデータを要求する際に使用する必要のあるデータベーススキーマの名前を指定します。デフォルトはありません。
watt.server.audit.schemaName が設定されていない場合、Integration Server はデータベースメタデータを取得せず、WMSERVICECUSTOMFLDS.STRINGVALUE 列および WMSERVICEACTIVITYLOG.FULLMESSAGE 列の長さがそれぞれ 512 および 1024 であると仮定します。

注:
一部のデータベースでは大文字と小文字を区別します。watt.server.audit.schemaName の値を指定するときは、スキーマ名の大文字と小文字がデータベースで必要なスキーマ名と一致する必要があります。

注:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.stdout.fieldDelimiter

監査ログがコンソールに書き込まれるとき (STDOUT) にログエントリ内の監査ログフィールドを区切る文字列を指定します。デフォルト値は「||」 (二重縦線記号) です。watt.server.audit.stdout.fieldDelimiter と watt.server.audit.stdout.recordDelimiter は異なる値にする必要があります。区切り文字シーケンスを指定する場合、ログエントリに含まれる可能性が高い文字シーケンスは使用しないでください。

環境変数 SAG_IS_AUDIT_STDOUT_LOGGERS を使用して STDOUT に書き込むよう監査ログを設定できます。設定変数の詳細については、1283 ページの「環境変数」を参照してください。

注:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.stdout.recordDelimiter
監査ロガーがコンソールに書き込まれるとき (STDOUT) に監査ログレコードを区切る文字列を指定します。デフォルト値は「&&」 (二重アンパサンド) です。watt.server.audit.stdout.fieldDelimiter と watt.server.audit.stdout.recordDelimiter は異なる値にする必要があります。区切り文字シーケンスを指定する場合、ログエントリに含まれる可能性が高い文字シーケンスは使用しないでください。

環境変数 SAG_IS_AUDIT_STDOUT_LOGGERS を使用して STDOUT に書き込むよう監査ログを設定できます。設定変数の詳細については、1283 ページの「環境変数」を参照してください。

注:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.um.sessionPool.min
Universal Messaging セッションプールのセッションの最小数を指定します。Integration Server は、監査ロガーによって使用される各 Universal Messaging 接続エイリアスの個別のセッションプールを保持します。このパラメータの値は、0 以上、watt.server.audit.um.sessionPool.max の値以下の整数値にする必要があります。デフォルト値は 2 です。

無効な値を指定すると、Integration Server は起動時に警告メッセージをログに記録し、実行時にはデフォルト値を使用します。ただし、Integration Server はデフォルト値で指定された無効な値を上書きして永続化することは行いません。

注:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.um.sessionPool.max
Universal Messaging セッションプールのセッションの最大数を指定します。Integration Server は、監査ロガーによって使用される Universal Messaging 接続エイリアスの個別のセッションプールを保持します。このパラメータの値は、0 よりも大きく、watt.server.audit.um.sessionPool.min の値以上の整数にする必要があります。デフォルト値は 10 です。

無効な値を指定すると、Integration Server は起動時に警告メッセージをログに記録し、実行時にはデフォルト値を使用します。ただし、Integration Server はデフォルト値で指定された無効な値を上書きして永続化することは行いません。

注:

このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.audit.um.sessionPool.retryInterval
Universal Messaging サーバで監査ロガーがキューをフェイルファストモードにした後に、Integration Server がセッションの再確立を試行する間隔 (秒単位) を指定します。監査ロガーは、監査ログキューを含む Universal Messaging サーバへの接続を確立できない場合、キューをフェイルファストモードにします。0 (ゼロ) よりも大きな整数を指定します。デフォルトは 30 秒です。

無効な値を指定すると、Integration Server は実行時にデフォルト値を指定します。ただし、Integration Server はデフォルト値で指定された無効な値を上書きして永続化することは行いません。

注:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.auditDocIdField
標準的な方法でドキュメントを識別し、ログ表示で一貫性のあるビジネスコンテキストを提供するためのカスタムドキュメント ID 値を指定します。一部のドキュメントは、webMethods Broker の WmLogUtil を通じてドキュメントデータベースに記録されます。また、一部のドキュメントは、たとえばサーバ障害の発生時やトリガーの再試行回数が超過した場合などに、Integration Server 内のさまざまなコンポーネントを通じて記録されます。この結果、ドキュメントモニタを表示する際、数値のドキュメント ID を使用して記録されるドキュメントもあれば、長い 16 進ストリングのドキュメント ID を使用して記録されるドキュメントもあります。ユーザが指定したカスタムドキュメント ID の値は、ドキュメントログ ID の作成に使用されます。BrokerEvent.getEventId() 値 (元のドキュメント ID の動作) に代わって、この値が使用されます。この値は、Broker の Unicode ストリング形式で指定する必要があります。128 文字を超える部分は切り捨てられます。この拡張設定が指定されていない場合は、元のドキュメント ID 動作が適用されます。この拡張設定が存在するが未定義 (NULL) の場合、_env.uuid の値が指定されていればこれが使用されます。_env.uuid が指定されていない場合は、元のドキュメント ID 動作が適用されます。ドキュメントログの詳細については、『*Administering webMethods Broker*』を参照してください。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.auth.cache.capacity
Integration Server によって認証キャッシュに格納される、ユーザ名とパスワードの組み合わせの数を指定します。デフォルト値は `250` です。

watt.server.auth.cache.enabled
認証キャッシュを有効にするかどうかを指定します。「true」に設定した場合、認証キャッシュは有効化されます。デフォルト値は `true` です。

watt.server.auth.cache.timeout
各キャッシュエントリがアイドル状態になってから、Integration Server が認証キャッシュからエントリを削除するまでの時間をミリ秒単位で指定します。デフォルトは `300000` (5 分) です。

注:
認証キャッシュから削除されたエントリは、次回クレデンシャルが正常に認証されたときに再度追加されます。

注:
ユーザがパスワードを変更し、新しいパスワードでのログインに成功すると、Integration Server は旧パスワードを認証キャッシュから削除します。

watt.server.auth.checkWithSession
Integration Server が認証ヘッダーとセッション ID 付きの Cookie ヘッダーの両方を含む HTTP 要求を受信した際、認証ヘッダーで識別されたユーザが Cookie ヘッダーで定義されているセッションの所有者であることを Integration Server でチェックする必要があるかどうかを、このパラメータで指定します。「`true`」(デフォルト) に設定すると、Integration Server は、認証ヘッダーのユーザが Cookie ヘッダーで識別されている所有者であるかどうかを確認します。ヘッダーが一致しない場合、Integration Server は、次のメッセージを含む HTTP 401 ステータスコードを戻します。

要求で識別されたユーザは、要求されたセッションを所有していません。

重要:
watt.server.auth.checkWithSession を「`false`」に設定した場合、Integration Server は HTTP 要求に対して、不一致のクレデンシャルを使用して Integration Server のサービスにアクセスすることを許可します。その結果、アプリケーションがリスクにさらされる場合があります。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.auth.oauth.accessToken.useHeaderFields
受信 HTTP/S 要求のヘッダーフィールドに access_token が含まれるときに Integration Server が OAuth 認証を実行するかどうかを指定します。「`true`」を指定すると、OAuth 認証を実行します。「`false`」を指定すると、OAuth 認証をスキップします。デフォルトは「true」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.auth.oauth.accessToken.useQueryParameters
受信 HTTP/S 要求のクエリーパラメータに access_token が含まれるときに Integration Server が OAuth 認証を実行するかどうかを指定します。「`true`」を指定すると、OAuth 認証を実行します。「`false`」を指定すると、OAuth 認証をスキップします。デフォルトは「`true`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.auth.samlResolver

Integration Server で My webMethods Server ユーザの妥当性検査に使用される、My webMethods Server の SAML アーティファクトリゾルバエンドポイントの URL を指定します。セントラルユーザ管理を設定すると共に、このプロパティの値を使用すると、My webMethods Server ユーザがシングルサインオンできるようになります。詳細については、556 ページの「MWS シングルサインオンのリソース設定」を参照してください。

注:
Integration Server Administrator の [**設定**] > [リソース] ページで [**MWS SAML リゾルバ URL**] フィールドの値を指定すると、Integration Server によって、このプロパティが server.cnf ファイルに追加されます。.

watt.server.auth.session.retainJaasSubject
Integration Server で、認証クレデンシャルをセッションの一部として保持する必要があるかどうかを指定します。「`true`」に設定すると、セッションが期限切れになるまで Integration Server は認証クレデンシャルを保持します。「`false`」(デフォルト) に設定すると、Integration Server は、クライアントからの最初の要求を処理した後、セッションからの認証クレデンシャルを削除します。

watt.server.auth.skipForMediator
Integration Server が、webMethods API Gateway からネイティブサービスへの要求の妥当性検査を行うかどうかを指定します。このパラメータを「`true`」に設定した場合、Integration Server は API Gateway 要求の認証をスキップします。「`false`」(デフォルト) に設定すると、Integration Server はすべての API Gateway 要求を認証します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.autodeploy.enabled
Microservices Runtime でパッケージの自動展開が有効か無効かを指定します。「true」に設定すると、パッケージの自動展開は有効化されます。「false」に設定すると、パッケージの自動展開は無効化されます。デフォルトは「false」です。

注:
自動展開機能は、Microservices Runtime ではデフォルトで使用可能です。Integration Server で自動展開機能を使用するには、Integration Server に追加ライセンスが必要です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Microservices Runtime を再起動する必要があります。

watt.server.autodeploy.interval
Microservices Runtime で自動展開システムタスクを実行する間隔を分単位で指定します。自動展開システムタスクは、自動展開される必要があるパッケージの自動展開フォルダを監視します。1 ～ 1440 までの値を指定します。デフォルトは 5 です。

注:
自動展開機能は、Microservices Runtime ではデフォルトで使用可能です。Integration Server で自動展開機能を使用するには、Integration Server に追加ライセンスが必要です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Microservices Runtime を再起動する必要があります。

watt.server.autodeploy.alwaysUseHotDeployment
Microservices Runtime がパッケージの自動展開に常にホット展開を使用するかどうかを指定します。パッケージの自動展開にホット展開を使用する場合は、「true」に設定します。パッケージの自動展開にホット展開を使用しない場合は、「false」に設定します。デフォルトは「false」です。

注:
自動展開機能は、Microservices Runtime ではデフォルトで使用可能です。Integration Server で自動展開機能を使用するには、Integration Server に追加ライセンスが必要です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Microservices Runtime を再起動する必要があります。

watt.server.broker.producer.multiclient
デフォルトクライアントのセッション数を指定します。デフォルトクライアントは、Broker にドキュメントをパブリッシュするときや、デフォルトクライアントにデリバーされたドキュメントを抽出するときに Integration Server が使用する Broker クライアントです。このパラメータを 1 より大きい値に設定した場合、Integration Server では、Broker へのドキュメントのパブリッシュに使用するために、*clientPrefix*_DefaultClient_MultiPub という名前の新しいマルチセッションの共有状態 Broker クライアントが作成されます。パブリッシュクライアントを複数セッションで使用すると、複数のスレッドがドキュメントを同時にパブリッシュできるため、パフォーマンスが向上する場合があります。デフォルトは 1 セッションです。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.broker.replyConsumer.fetchSize
Integration Server が Broker から一度に抽出する応答ドキュメント数を指定します。Integration Server で呼び出しごとに抽出する応答ドキュメントを増やすと、Integration Server で Broker に対して行われる呼び出し数を削減できます。Integration Server では、すべての応答ドキュメントがメモリに維持されます。Integration Server で一度に抽出するドキュメント数を減らすことにより、応答ドキュメントに使用されるメモリ量を削減できます。デフォルトは 5 ドキュメントです。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.broker.replyConsumer.multiclient
要求/応答クライアントのセッション数を指定します。要求/応答クライアントは、Broker が Integration Server に要求ドキュメントを送信するとき、および Broker から応答ドキュメントを抽出するときに使用する Broker クライアントです。要求/応答クライアントのセッション数を増やすと、複数の要求と応答を同時に送信および抽出できるため、パフォーマンスが向上する可能性があります。デフォルトは 1 セッションです。

注:

webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.broker.replyConsumer.sweeperInterval
Integration Server で内部メールボックスをスイープして、パブリッシュ済み要求への期限切れの応答を削除する間隔 (ミリ秒単位) を指定します。間隔の長さは、期限切れの応答によって消費されるメモリ量と待機要求に対する応答の抽出との間でバランスをとる必要があります。Integration Server では、期限切れの応答のエージングと削除には 1 つのバックグラウンドスレッドが使用され、待機要求の応答の抽出には複数のバックグラウンドスレッドが使用されます。スイーパスレッドが期限切れの応答を削除するときに、そのスイーパスレッドは応答の抽出を試行するスレッドをブロックします。スイーパ間隔が短すぎると、スイーパスレッドが頻繁に実行されるために他のバックグラウンドスレッドが応答を頻繁に抽出できなくなり、パフォーマンスが低下することがあります。スイーパ間隔が長すぎると、期限切れの応答がメモリを長期間消費するため、メモリ使用率が高くなる場合があります。デフォルトは 30000 ミリ秒 (30 秒) です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.brokerTransport.dur
キープアライブメッセージを Integration Server に送信する前に Broker が待機するアイドル時間の秒数を指定します。watt.server.brokerTransport.max プロパティで指定されている時間内に Integration Server が応答しない場合、Broker はもう 1 つのキープアライブメッセージを Integration Server に送信します。引き続き Integration Server からの応答がなければ、`watt.server.brokerTransport.ret` プロパティで指定されている再試行回数の上限に達するまで、Broker はキープアライブメッセージの送信を続行します。それでも Integration Server がキープアライブメッセージに応答しなければ、Broker は明示的に Integration Server を切断します。watt.server.brokerTransport.dur の値は、ゼロ以上かつ 2147483647 未満の整数にする必要があります。デフォルトは 60 秒です。

サーバパラメータを使用して Broker でのキープアライブ設定を構成する方法の詳細については、284 ページの「キープアライブモードのサーバ設定パラメータ」を参照してください。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.brokerTransport.max
Integration Server がキープアライブメッセージに応答するのを Broker が待機する秒数を指定します。この値は、0～2147483647 の整数にする必要があります。デフォルトは 60 秒です。

サーバパラメータを使用して Broker でのキープアライブ設定を構成する方法の詳細については、284 ページの「キープアライブモードのサーバ設定パラメータ」を参照してください。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.brokerTransport.ret
応答のない Integration Server を切断する前に Broker がキープアライブメッセージを再送信する回数を指定します。この値は、1～2147483647 の整数にする必要があります。デフォルトの試行回数は 3 回です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.cache.prefetchUser
キャッシュサービスエントリをプリフェッチする権限を持つユーザを指定します。デフォルトは `Administrator` です。

watt.server.cache.flushMins
サーバが期限切れのキャッシュエントリを削除してキャッシュサービスエントリをプリフェッチするためにキャッシュを何分ごとにスイープするかを指定します。デフォルトは 10 分です。

注:
この設定パラメータは、サービス結果のキャッシング機能に適用されます。Terracotta Ehcache で提供されるキャッシング機能には影響しません。

watt.server.cache.gcMins
ガーベッジコレクションを実行するためにサーバがキャッシュを何分ごとにスイープするかを指定します。デフォルトは 60 分です。

注:
この設定パラメータは、サービス結果のキャッシング機能に適用されます。Terracotta Ehcache で提供されるキャッシング機能には影響しません。

watt.server.cache.maxEntriesInCache
Integration Server クラスタで使用される分散システムキャッシュの [**キャッシュの最大エントリ数**] プロパティのデフォルト値を指定します。デフォルトは -1 です。これは、Integration Server に [**キャッシュの最大エントリ数**] プロパティのデフォルト値が設定されていないことを示します。Integration Server に [**キャッシュの最大エントリ数**] プロパティのデフォルト値を設定する場合、0 より大きい値を指定する必要があります。

watt.server.cachemanager.connectTimeout
Terracotta Server Array の URL リストで指定されたサーバに接続するまでに、キャッシュマネージャが待機する時間をミリ秒単位で指定します。デフォルトは 60000 ミリ秒 (60 秒) です。

注:
watt.server.cacheManager.connectTimeout 値に関係なく、キャッシュマネージャは Terracotta Server Array のサーバへの接続を最大で 300 秒間待機します。たとえば、このプロパティを 60000 に設定した場合、キャッシュマネージャは使用可能な任意のサーバへの接続を 60000 ミリ秒待機します。任意のサーバへの接続が 60000 ミリ秒で確立されない場合、例外がスローされ、Integration Server は watt.server.cluster.action.errorOnStartup パラメータで指定されたアクションを実行します。

watt.server.cachemanager.logsDirectory
Terracotta Ehcache がログファイルを書き込む場所を指定します。このフォルダのデフォルト値は、 *Integration Server_directory* \instances*instance_name*\logs\tc-client-logs です。

watt.server.cachemanager.parallelThreads

Integration Server の起動時にキャッシュマネージャを初期化するときに使用する最大スレッド数を指定します。

サーバに多数のキャッシュマネージャ、分散キャッシュマネージャ、または多数のキャッシュを含むキャッシュマネージャが存在する場合、起動時の登録プロセスに時間がかかることがあります。これを回避するために、watt.server.cachemanager.parallel.threads プロパティを設定して、キャッシュマネージャの初期化時に使用するスレッドの数を増やすことができます。

Integration Server でキャッシュマネージャを順次に初期化する場合、値を 1 に設定します。Integration Server で並行してキャッシュマネージャを初期化する場合は、値を 2 以上に設定します。値を 10 以下にすることをお勧めします。デフォルト値は 5 です。

watt.server.centralUsers.shutdownOnError
Central Users コンポーネントでエラーが発生した場合に、コンポーネントをシャットダウンするかどうかを指定します。このプロパティを「`true`」に設定した場合、Central User コンポーネントでエラーが発生するとコンポーネントがシャットダウンされます。Central Users コンポーネントがシャットダウンした場合は、Integration Server を再起動する必要があります。このプロパティをデフォルトの「`false`」に設定した場合、Central User コンポーネントでエラーが発生した場合でもコンポーネントはシャットダウンされません。

watt.server.cgi.cache
応答をフォーマットするために CGI コンテンツハンドラを使用するときに、出力パイプラインで応答のすべてのエレメントを表示するかどうかを指定します。このプロパティを「`false`」に設定すると、出力パイプラインで応答のすべてのエレメントが表示されます。このプロパティを「`true`」に設定すると、応答にストリング以外で NULL 以外の同じ値を持つ複数のエレメントが含まれるときは、出力パイプラインで値の最初の発生のみが表示されます。後続の発生ではそれぞれ、出力にストリング *`*ObjectRef(className)*`* が含まれます。デフォルトは「`false`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.cgi.unicode
CGI コンテンツハンドラで Unicode エンコーディングを使用するかどうかを指定します。このプロパティを「`false`」に設定すると、CGI コンテンツハンドラは、watt.server.netencoding プロパティで指定されたエンコーディングを使用して応答をエンコードします。このプロパティを「`true`」に設定すると、CGI コンテンツハンドラは Unicode を使用して応答をエンコードします。デフォルトは「`false`」です。

watt.server.checkAclsInternally
サービスがクライアントまたはトリガーによって直接呼び出された場合、およびサービスが他のサービスから呼び出された場合に、Integration Server で ACL チェックを実行するかどうかを指定します。「true」に設定した場合、Integration Server では、サービスの [実行 ACL の適用] プロパティの値に関係なく、サービスの実行 ACL が常にチェックされます。「false」に設定した場合、サービスの実行時に Integration Server で ACL チェックが実行されるかどうかは、[実行 ACL の適用] プロパティの値によって決まります。デフォルトは「false」です。

watt.server.checkPath.restorePipelineFromFile

Integration Server が、pub.flow:restorePipelineFromFile サービスに指定された *filename* 入力パラメータの値がデフォルトのパイプラインディレクトリにあるか、または、ファイルアクセスコントロール設定ファイル (fileAccessControl.cnf) の allowedReadPaths パラメータにあるかを確認するどうかを指定します。このパラメータを「true」に設定した場合、Integration Server は *filename* で指定されたパスまたはディレクトリがデフォルトのパイプラインディレクトリに含まれているか、または allowedReadPaths パラメータにあるかを確認します。ファイルが許可リストにない場合、サービスは ServiceException で終了します。
watt.server.checkPath.restorePipelineFromFile パラメータを「false」に設定した場合、Integration Server は指定されたファイル名がデフォルトのパイプラインディレクトリにあるか、または allowedReadPaths パラメータに示されているかを確認しません。デフォルトは、より安全なオプションである「true」です。

watt.server.checkPath.savePipelineToFile
Integration Server が、pub.flow:savePipelineToFile サービスに入力された *filename* 入力パラメータの値がデフォルトのパイプラインディレクトリにあるか、またはファイルアクセスコントロール設定ファイル (fileAccessControl.cnf) の allowedWritePaths パラメータにあるかを確認するどうかを指定します。このパラメータを「true」に設定した場合、Integration Server は *filename* で指定されたパスまたはディレクトリがデフォルトのパイプラインディレクトリにあるか、または allowedWritePaths パラメータに含まれているかを確認します。ファイルが allowedWritePaths リストにない場合、サービスは ServiceException で終了します。
watt.server.checkPath.savePipelineToFile パラメータを「false」に設定した場合、Integration Server は指定されたファイル名がデフォルトのパイプラインディレクトリにあるか、または allowedWritePaths パラメータに示されているかを確認しません。デフォルトは、より安全なオプションである「true」です。

watt.server.checkWhitelist
Integration Server がホワイトリストを使用して、デシリアライズに使用できるクラスのリストをフィルタするかどうかを指定します。このパラメータを true に設定すると、ホワイトリストフィルタ処理は有効になります。Integration Server では、クラスがホワイトリストにある場合にのみ、Java オブジェクトをデシリアライズします。クラスがホワイトリストにない Java オブジェクトが見つかると、Integration Server は ClassNotFoundException をスローします。デフォルトでは、watt.sever.checkWhitelist は「true」に設定されます。これは、ホワイトリストフィルタ処理が有効であることを示します。

ホワイトリストフィルタ処理の詳細については、977 ページの「Integration Server におけるホワイトリストフィルタ処理 」を参照してください。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.circuitBreaker.threadPoolMax
サーキットブレーカーが設定されたサービスの実行に使用されるサーキットブレーカースレッドプールにサーバが保持するスレッドの最大数を指定します。この最大スレッド数に達した場合、サーバは、現在のサービスが完了してスレッドがサーキットブレーカースレッドプールに戻されるまで待機してから、サーキットブレーカーが設定された次のサービスを実行します。0 より大きく、watt.server.circuitBreaker.threadPoolMin の値以上の値を指定する必要があります。デフォルトは 75 です。

> **重要:**

このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

注:
サーキットブレーカー機能は Microservices Runtime にあるサービスでデフォルトで利用できます。Integration Server でサーキットブレーカー機能を使用するには、Integration Server に追加ライセンスが必要です。

watt.server.circuitBreaker.threadPoolMin
サーキットブレーカーが設定されたサービスの実行に使用されるサーキットブレーカースレッドプールにサーバが保持するスレッドの最小数を指定します。サーバ起動時のサーキットブレーカースレッドプールには、初期値としてここで指定された最小スレッド数が設定されます。watt.server.circuitBreaker.threadPoolMax で指定した最大スレッド数に達するまで、サーバは必要に応じてスレッドをプールに追加します。0 (ゼロ) 以上で、watt.server.circuitBreaker.threadPoolMax の値以下の値を指定する必要があります。デフォルトは 10 です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

注:
サーキットブレーカー機能は Microservices Runtime にあるサービスでデフォルトで利用できます。Integration Server でサーキットブレーカー機能を使用するには、Integration Server に追加ライセンスが必要です。

watt.server.classloader.pkgpriority
Integration Server でパッケージ情報のないクラスファイルをロードするときのパッケージのスキャン順序を指定します。順序を指定しなかった場合、Integration Server では、クラスのロード時に Packages ディレクトリ内のすべてのパッケージが 1 つずつスキャンされます。そのため、クラスのロードが遅れることがあります。たとえば、Java クラスへの参照が含まれているトリガーには、パッケージ情報はありません。Integration Server では、特定のクラスファイルを見つけるために、すべてのパッケージがランダムな順序でスキャンされます。Integration Server で最初にスキャンするパッケージを指定すると、クラスのロードにかかる時間を短縮できる場合があります。

watt.server.classloader.pkgpriority プロパティを使用して、クラスローダで最初にロードするパッケージをカンマ区切りのリストで指定します。このプロパティの構文は次のとおりです。

```
watt.server.classloader.pkgpriority=packageName, packageName
```

クラスのロードの詳細については、963 ページの「Integration Server におけるクラスロード」を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.clientTimeout
再使用されたアイドル状態のユーザセッションがタイムアウトするまでの時間 (分) を指定します。デフォルトは `10` です。

watt.net.http.clientSession.idleTimeout パラメータは、新規作成された送信 HTTP/S セッションがタイムアウトになり、セッションスイーパによって削除される前に、そのセッションがアイドル状態でいることができる時間をミリ秒単位で決定します。

watt.server.clientTimeoutRedirect
ブラウザのセッションがタイムアウトになったときに Integration Server Administrator がどのように動作するかを指定します。このパラメータを「`true`」に設定すると、Integration Server Administrator は、セッションを再開するためのログインプロンプトを表示するオプションを含んだ空白ページにリダイレクトします。ログインが成功すると、ホームページが表示されます。デフォルト値である「`false`」に設定されている場合は、Integration Server Administrator は、バックグラウンドで現在のページのコンテンツを保持しつつ、ログインをプロンプトします。ログインが成功すると、操作は同じページから再開されます。

watt.server.cluster.action.errorOnStartup
起動時のエラーで Integration Server がクラスタを結合できないときの Integration Server の対応方法を指定します。次のいずれかを選択します。

値	説明
`shutdown`	Integration Server で起動時にクラスタを結合できないようなエラーが発生した場合、Integration Server は直ちにシャットダウンします。クラスタリング設定に変更を加えるときは、Integration Server をセーフモードで起動して設定を編集し、Integration Server を再起動する必要があります。 「`shutdown`」は、[**設定**] > [クラスタリング] > [クラスタ設定の編集] ページにある [**起動エラー時のアクション**] パラメータオプションの [**Integration Server をシャットダウン**] に対応します。
`standalone`	Integration Server で起動時にクラスタを結合できないようなエラーが発生した場合、Integration Server はクラスタが解除されたスタンドアロンの Integration Server として起動します。Integration Server は、受信ポートで要求の受信と要求の処理を継続します。Integration Server は分散キャッシュではなく、ローカルキャッシュを使用します。起動エラーの解決に必要な変更を加えるために、Integration Server を使用できます。Integration Server を再起動すると、Integration Server で Terracotta Server Array に接続できないような起動エラーが発生しない場合は、Integration Server によってクラスタが再結合されます。 これがデフォルト値です。 「`standalone`」は、[**設定**] > [クラスタリング] > [クラスタ設定の編集] ページにある [**起動エラー時のアクション**] パラメータオプションの [**スタンドアロン** Integration Server **として開始**] に対応します。
`quiesce`	Integration Server で起動時にクラスタを結合できないようなエラーが発生した場合、Integration Server はクラスタが解除されたスタンドアロンの Integration Server として休止モードで起動します。Integration Server が休止モードのときは、診断ポートおよび休止ポートだけが有効化されています。Integration Server は、受信ポートで要求を受信しません。ただし、起動エラーの修正に必要な変更を加えるために、Integration Server

値	説明
	および Integration Server Administrator を使用できます。エラーを修正したら、休止モードを終了します。休止モードを終了すると、Integration Server はすべてのキャッシュマネージャを起動し、クラスタを結合します。 休止モードを終了するときにクラスタを結合できないようなエラーが Integration Server で発生した場合、Integration Server は休止モードを再度開始しません。代わりに、Integration Server はクラスタが解除されたスタンドアロンサーバとして実行します。Integration Server は、クラスタを結合しようとしたときに発生したすべてのエラーをレポートに取り込みます。このレポートは、サーバの休止モードを終了した後で Integration Server Administrator に表示されます。レポートおよびエラーログは、根本的な設定エラーを修正するのに役立ちます。その後にクラスタを再結合するには、Integration Server を再起動するか、または休止モードをいったん開始してから終了します。 「quiesce」オプションを使用するには、Integration Server 用に休止ポートを設定する必要があります。休止ポートを設定せずに「quiesce」オプションを選択した場合、Integration Server で起動時にクラスタを結合できないようなエラーが発生すると、Integration Server は休止モードを開始せずにシャットダウンします。 「quiesce」は、[**設定**] > [クラスタリング] > [クラスタ設定の編集] ページにある [**起動エラー時のアクション**] パラメータオプションの [**スタンドアロン** Integration Server **で休止モードに入る**] に対応します。 休止モードの詳細については、983 ページの「保守のためのサーバの休止」を参照してください。

watt.server.cluster.action.errorOnStartup パラメータにより、Integration Server Administrator の **[設定] > [クラスタリング]** ページおよび **[設定] > [クラスタリング] > [クラスタ設定の編集]** ページの [**起動エラー時のアクション**] パラメータで指定された値が表示されます。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.cluster.aliasList
クラスタ内にあるリモートの Integration Server のエイリアスをカンマ区切りのリストで指定します。Integration Server は、トリガー管理の変更内容を他のクラスタノードに適用するリモート呼び出しを実行するときに、このリストを使用します。このプロパティを設定すると、トリガーの管理作業を実行するときに、[**メッセージング**] > [**webMethods トリガー**] > [クラスタの表示] ページが表示され、[**クラスタ全体に変更内容を適用**] チェックボックスを使用できるようになります。

このパラメータは、webMethods クラスタリングを使用する場合にのみ適用できます。詳細については、『*webMethods Integration Server Clustering Guide*』を参照してください。

watt.server.cluster.aware

サーバがクラスタに参加するかどうかを指定します。「true」の値は、クラスタリングがサーバに対して有効になっていることを示します。デフォルトは「false」です。

watt.server.cluster.aware パラメータの値は、Integration Server Administrator の [**設定**] > [クラスタ] ページの [**クラスタ機能の状態**] フィールドの値に関連付けられています。

このパラメータは、クラスタでこの Integration Server を使用する場合にのみ適用できます。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

注:
watt.server.cluster.aware を「true」に設定した場合は、watt.server.cluster.tsaURLs パラメータおよび watt.server.cluster.name パラメータにも値を指定する必要があります。

watt.server.cluster.name
Integration Server が属しているクラスタの名前を指定します。Integration Server がクラスタに属していない場合は、このパラメータは空です。

クラスタ名を指定する場合は、以下の点に留意してください。

- クラスタ名にピリオド (.) は使用できません。クラスタ設定を保存すると、Integration Server は名前に含まれるピリオドをアンダースコアに変換します。
- クラスタ名は 32 文字以下にします。32 文字を超えると、Integration Server は入力された名前の最初の 20 文字を使用し、残りの文字をハッシュにします。

watt.server.cluster.name パラメータにより、Integration Server Administrator の [**設定**] > [クラスタ] ページおよび [**設定**] > [クラスタ] > [クラスタ設定の編集] ページの [**クラスタ名**] パラメータで指定された値が表示されます。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

注:
watt.server.cluster.aware を「true」に設定した場合は、watt.server.cluster.tsaURLs パラメータおよび watt.server.cluster.name パラメータにも値を指定する必要があります。

watt.server.cluster.SessTimeout
サーバが、非アクティブなセッションオブジェクトを削除する前にクラスタストア内に残しておく時間 (分) を指定します。デフォルトは 60 です。

このパラメータは、webMethods Integration Server クラスタリングを使用する場合にのみ適用できます。詳細については、『*webMethods Integration Server Clustering Guide*』を参照してください。

watt.server.cluster.tsaURLs
クラスタで分散システムキャッシュに対して使用される Terracotta Server Array の URL のカンマ区切りリスト。このパラメータは、Integration Server がクラスタに属する場合にのみ適用できます。

watt.server.cluster.tsaURLs パラメータは、Integration Server Administrator の **[設定] > [クラスタリング]** ページおよび **[設定] > [クラスタリング] > [クラスタ設定の編集]** ページの [**Terracotta Server Array URL**] パラメータに関連付けられています。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

注:
watt.server.cluster.aware を「`true`」に設定した場合は、watt.server.cluster.tsaURLs パラメータおよび watt.server.cluster.name パラメータにも値を指定する必要があります。

watt.server.coder.bincoder.trycontextloaderfirst
Integration Server でパイプラインをエンコードまたはデコードするときに、現在実行されているスレッドに対して、Integration Server でクラスローダを使用する前にコンテキストローダを使用するかどうかを指定します。参照されるクラスが特定のパッケージに属している場合、最初にコンテキストローダを使用した方が処理が速くなることがあります。デフォルトは「false」です。

watt.server.coder.responseAsXML
Integration Server が REST API 要求に対する XML 応答を受信する方法を指定します。このプロパティを「`true`」に設定した場合、Integration Server は適切な XML フォーマットで任意の REST API 要求に対する XML 応答を受信します。このプロパティを「`false`」に設定した場合、Integration Server は IData オブジェクトを含む変換された XML フォーマットで、任意の REST API に対する XML 応答を受信します。デフォルトは「`true`」です。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.commonmessaging.connection.retryPeriod
Integration Server で MQTT サーバへの接続が失敗した場合の接続試行間の待機時間を秒単位で指定します。デフォルトは 20 秒です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.commonmessaging.trigger.monitoringInterval
Integration Server が MQTT トリガーに対するリソース監視サービスを実行する間隔を秒単位で指定します。リソース監視サービスとは、MQTT トリガーサービスによって使用されるリソースの可用性をチェックするためにユーザが作成するサービスです。再試行の試みがすべて失敗した場合、Integration Server は MQTT トリガーを停止した後に、システムタスクをスケジュールして MQTT トリガーに割り当てられているリソース監視サービスを実行します。デフォルトは 60 秒です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.commonmessaging.trigger.restartTaskRetryCount
MQTT トリガーを自動的に開始するトリガーの再開始タスクの最大試行数を指定します。トリガーによって使用される MQTT 接続エイリアスの起動時に、Integration Server は MQTT トリガーの最初の開始を試行します。エイリアスの起動時に MQTT トリガーが開始に失敗した場合、Integration Server はトリガーの再開始タスクを使用して失敗した MQTT トリガーの開始を再試行します。再開始タスクは、エイリアスを開始するのに使用されたものとは別の

スレッドを使用し、watt.server.commonmessaging.trigger.restartTaskRetryInterval パラメータによって指定された間隔で実行されます。また、実行時にトリガーに障害が発生した場合も、Integration Server は再開始タスクを使用します。たとえば、MQTT 接続エイリアスが利用できなくなった場合、MQTT トリガーは最終的に失敗します。

再開始タスクを使用するには、1 以上の正の整数を指定する必要があります。デフォルトの再試行回数は 6 回です。Integration Server で MQTT トリガーを自動的に再開始しない場合は、watt.server.commonmessaging.trigger.restartTaskRetryInterval を 0 に設定します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.commonmessaging.trigger.restartTaskRetryInterval
トリガーの再開始タスクが MQTT トリガーの再開始の試行間に待機する秒数を指定します。正の整数を指定する必要があります。デフォルトは 30 秒です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.commonmessaging.trigger.reuseSession
MQTT トリガーの複数のインスタンスが Integration Server の同じセッションを使用するかどうかを指定します。このプロパティを「true」に設定した場合、MQTT トリガーは共有セッションを使用します。MQTT トリガーによって実行される各トリガーサービスが同じセッション ID を使用します。このプロパティを「false」に設定した場合、Integration Server は MQTT トリガーのインスタンスごとに新しいセッションを使用します。MQTT トリガーでセッションを再使用することにより、パフォーマンスが向上する可能性があります。デフォルトは「false」です。

watt.server.commonmessaging.trigger.stopRequestTimeout
MQTT トリガーが無効になってから MQTT トリガーによるメッセージの処理を強制的に停止するまでの Integration Server の最大待機時間を秒単位で指定します。最小値は 0 です。デフォルトは 3 秒です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.compile
javac -classpath {0} -d {1} {2} など、Designer を使用して開発された Java サービスのコンパイルに Integration Server が使用するコンパイラコマンドを指定します。このコンパイラコマンドも、jcode ユーティリティから使用されます。このプロパティを省略したか、または空の場合、サーバでは Java サービスのコンパイルに JVM 内部 Java コンパイルツールが使用されます。

注:
Integration Server で別のコンパイラを使用するように watt.server.compile プロパティを使用していて、そのコンパイラがエラーや警告を標準エラーではなく標準出力に返す場合は、コンパイラのエラーおよび警告を Java 開発者が読み取ったり表示したりできるように watt.server.compile.readFromStdErr を「`false`」に設定します。

watt.server.compile.exitOnError
jcodeユーティリティが処理を失敗したときに、停止するかどうかを指定します。「`true`」に設定すると、jcodeユーティリティは失敗したパッケージで処理を停止します。「`false`」(デフォルト) に設定すると、jcode ユーティリティは失敗しても処理を続行します。

watt.server.compile.readFromStdErr
Integration Server がコンパイラのエラーおよび警告を読み取る場所を指定します。Software AG webMethods プラットフォームには javac が含まれています。javac は Designer で jcode ユーティリティを使用して Java サービスをコンパイルするための Java ツールです。javac コンパイラは、コンパイル時に発生したエラーまたは警告に関する情報を標準エラーに返します。Integration Server で別のコンパイラを使用するように watt.server.compile プロパティを使用していて、そのコンパイラがエラーや警告を標準エラーではなく標準出力に返す場合は、コンパイラのエラーおよび警告を Java 開発者が読み取ったり表示したりできるように watt.server.compile.readFromStdErr を「`false`」に設定します。Integration Server がコンパイラのエラーおよび警告を標準エラーから読み取るようにするには、watt.server.compile.readFromStdErr を「`true`」に設定します。デフォルトは「true」です。

watt.server.compile.unicode
javac -encoding Unicode -classpath {0} -d {1} {2} など、Unicode エンコーディングで格納されている Java サービスのコンパイルに Integration Server が使用するコンパイラコマンドを指定します。javac -encoding Unicode -classpath {0} -d {1} {2} の設定は Oracle JDK コンパイラでも有効です。このコンパイラコマンドも、jcodeユーティリティから使用されます。このプロパティを省略したか、または空の場合、サーバでは Java サービスのコンパイルに JVM 内部 Java コンパイルツールが使用されます。

watt.server.content.type.default
Content-Type ヘッダーを含まない HTTP 要求に使用するコンテンツタイプを指定します。デフォルト値は `application/octet-stream` です。

> **注:**
> watt.server.content.type.default の値の指定がなく、Content-Type ヘッダーが空の場合、Integration Server は `application/octet-stream` のデフォルト値を割り当てます。この場合は、application/octet-stream 用のコンテンツハンドラを作成する必要があります。

Integration Server で別の Content-Type を使用するように指定するには、`text/xml` または `text/html` など、登録されている別のコンテンツタイプに watt.server.content.type.default の値を変更します。watt.server.content.type.default の値を登録されていないコンテンツタイプに設定した場合、Integration Server では `text/html` の Content-Type ヘッダーとして要求を処理します。

このプロパティの変更は即座に反映されます。

watt.server.content.type.mappings
ワイルドカードが設定されたコンテンツタイプから特定のコンテンツタイプへのマッピングを含め、コンテンツタイプの一対一マッピングを指定します。

このパラメータの使用方法の 1 つは、HTTP クライアント要求の承認ヘッダーフィールドで見つかったワイルドカードを特定のコンテンツタイプにマッピングすることです。承認ヘッダーフィールドには、クライアントが応答で受け入れることになる 1 つまたは複数のコンテンツタイプが指定されています。Integration Server は、許可されているコンテンツタイプに基づいて送信コンテンツハンドラを選択します。

このパラメータのもう 1 つの使用方法は、受信要求の特定のコンテンツタイプに使用するコンテンツハンドラを指定することです。

このパラメータの構文は次のとおりです。

watt.server.content.type.mappings=<*wildcard*> <*content-type*>,<*wildcard*> <*content-type*>,<*wildcard*> <*content-type*> ...

ここで、

- *wildcard* は text/* などのワイルドカードが設定されたコンテンツタイプまたはワイルドカード文字がないコンテンツタイプです。
- *content-type* はワイルドカードが設定されたコンテンツタイプまたはワイルドカード文字がない指定コンテンツタイプに使用する特定のコンテンツタイプです。

ワイルドカードで指定するコンテンツタイプと特定のコンテンツタイプはスペースで区切ります。<*wildcard*> <*content-type*> のペア間はカンマで区切ります。次の例について考えます。

`text/*` を `text/xml` に関連付けるには、次のように指定します。

```
watt.server.content.type.mappings=text/* text/xml
```

`text/*` を `text/xml` に関連付け、`multipart/*` を `multipart/related` に関連付けるには、次のように指定します。

```
watt.server.content.type.mappings=text/* text/xml,multipart/* multipart/related
```

Integration Server は承認ヘッダーを解析して、W3C ハイパーテキスト転送プロトコル仕様で定義されているコンテンツタイプまたはタイプの選択を試みます。

デフォルト設定は、`text/* text/xml` です。このパラメータを指定しなかった場合、承認ヘッダーフィールドにワイルドカードが指定されていれば、Integration Server は以下に示すように指定された主要コンテンツタイプに基づいてコンテンツハンドラを選択します。

承認ヘッダーフィールド	選択されるコンテンツハンドラ
text/*	XML
application/*	CGI
multipart/*	Multipart

watt.server.content.type.mappings パラメータを機能させるには、watt.server.http.useAcceptHeader プロパティを「true」に設定する必要があります。

Integration Server で受信要求に対して application/xml コンテンツハンドラの代わりに text/xml コンテンツハンドラを使用するために、watt.server.content.type.mappings を使用して application/xml コンテンツタイプを text/xml コンテンツタイプにマップするには、以下を指定します。

```
watt.server.content.type.mappings=application/xml text/xml
```

コンテンツハンドラの詳細については、953 ページの「Integration Server のコンテンツハンドラ 」を参照してください。

watt.server.control.controlledDeliverToTriggers.delayIncrementInterval

watt.server.control.controlledDeliverToTrigger.delays と関連して使用する間隔を指定します。これにより、トリガードキュメントストアが容量の 90% に達したときに、Integration Server

がローカルにドキュメントをパブリッシュするサービスに適用する遅延を決定します。デフォルトは 2000 です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.control.controlledDeliverToTriggers.delays
トリガードキュメントストアが容量の 90% 以上に達したときに、Integration Server がローカルドキュメントをパブリッシュするサービスを遅延させる秒数を、ミリ秒単位で指定する値のカンマ区切りリストを指定します。Integration Server は、トリガードキュメントストアが継続して容量の 90% 以上になっている期間に基づいて、遅延を変更します。デフォルト値は 500,2000,3000,3500,4000,4500,5000 です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.control.controlledDeliverToTriggers.pctMaxThreshold
Integration Server がローカルにパブリッシュされたドキュメントをデリバーするペースを落とす、トリガーキューのしきい値を指定します。このしきい値はトリガーキューの容量のパーセンテージとして表されます。たとえば 80 を指定した場合、トリガーキューの容量の 80% に達すると、Integration Server はローカルにパブリッシュされたドキュメントをトリガーキューにデリバーするペースを落とします。トリガーキューの容量が、指定したしきい値を下回ると、Integration Server はドキュメントのデリバーを通常のペースで再開します。デフォルトは 90 です。

watt.server.control.freeMemoryThreshold
Integration Server で、空きメモリが不足していることを示す警告を生成するしきい値をパーセンテージとして指定します。使用可能な空きメモリの割合がこのプロパティの値以下になると、Integration Server によってサーバログに警告が生成されます。有効な値は 0～100 です。デフォルトは 5 です。

watt.server.control.maxPersist
送信ドキュメントストアの容量を指定します。Integration Server は、設定済みの Broker が使用できないとき、パブリッシュされたドキュメントを送信ドキュメントストアに配置します。送信ドキュメントストア内のドキュメント数が、指定した容量に達すると、Integration Server はドキュメントをパブリッシュするサービスを実行しているスレッドをすべてブロックします。Integration Server は、Broker が使用可能になってから、ブロックしたスレッドの実行を再開します。このパラメータは 0 よりも大きい整数に設定してください。デフォルトは 500,000 ドキュメントです。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.control.maxPublishOnSuccess
サーバが一度にパブリッシュすることができるドキュメントの最大数を指定します。たとえば、ドキュメントの最大数を 100 に設定したとします。ServiceA は 10 個のドキュメントをパブリッシュします。ServiceB は 90 個のドキュメントをパブリッシュします。ServiceC は 5 個のドキュメントをパブリッシュします。この場合、ServiceA と ServiceB は同時にドキュメントをパブリッシュすることができます。ただし、ServiceA または ServiceB がこれらのド

キュメントのパブリッシュを完了する前に ServiceC がドキュメントのパブリッシュを開始すると、Integration Server は ServiceC に例外をスローします。これは、ServiceC がパブリッシュするドキュメント数が、一度にパブリッシュ可能なドキュメントの最大数を超えているためです。ServiceD が 125 個のドキュメントを発行しようとしても、最大数が 100 に設定されていれば、ServiceD はパブリッシュを実行するたびに例外を受け取ります。デフォルトは 50,000 ドキュメントです。

watt.server.control.memorySensorInterval
Integration Server が使用可能な空きメモリをチェックする間隔 (ミリ秒単位)。デフォルトは `300000` (5 分) です。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.control.serverThreadThreshold
Integration Server で、使用可能なスレッドが不足していることを示す警告を生成するしきい値を指定します。使用可能なサーバスレッドの割合がこのプロパティで指定された値を下回ると、Integration Server で「**使用可能なスレッド警告しきい値を超過しました**」という内容のジャーナルログメッセージが生成され、現在使用可能なスレッドの割合が示されます。このメッセージをジャーナルログで受け取ったら、スレッドの使用状況を調整してサーバスレッドを使用可能にすることができます。
デフォルトは 15% です。

> **注:**
> Integration Server Administrator の [リソースの設定の編集] ページの [**使用可能なスレッド警告しきい値**] フィールドを使用して、しきい値を設定することをお勧めします。使用可能なスレッド警告しきい値の設定の詳細については、127 ページの「サーバスレッドプールの管理」を参照してください。

watt.server.control.threadSensorInterval
Integration Server でスレッドの使用状況を監視する間隔をミリ秒単位で指定します。使用可能なサーバスレッドの割合が watt.server.control.serverThreadThreshold プロパティで指定されたしきい値を下回ると、Integration Server は使用可能なスレッドが不足していることを警告するジャーナルログメッセージを生成します。デフォルトは 2000 ミリ秒です。

watt.server.control.triggerInputControl.delayIncrementInterval
watt.server.control.triggerInputControl.delays と関連して使用する間隔を指定します。これにより、最後にポーリングされたときにキューが空の場合に Integration Server が Broker 上のトリガークライアントキューをポーリングする頻度を決定します。デフォルトは 10000 です。

> **注:**
> watt.server.control.triggerInputControl.delays または watt.server.control.triggerInputControl.delayIncrementInterval で指定された値を解析するときに例外が発生した場合、Integration Server は両方の設定パラメータをデフォルト値にリセットします。

> **注:**
> webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

Broker でトリガークライアントキューのポーリング頻度を制御する方法の詳細については、906 ページの「webMethods Messaging Triggerのポーリング要求の遅延」を参照してください。

watt.server.control.triggerInputControl.delays
Integration Server が Broker でのトリガークライアントキューのポーリングを遅延させる秒数を、ミリ秒単位で指定する値のカンマ区切りリストを指定します。Integration Server は、トリガークライアントキューが空になっている期間に基づいて、遅延を変更します。デフォルト値は 500,1000,1500,5000 です。

注:
watt.server.control.triggerInputControl.delays または watt.server.control.triggerInputControl.delayIncrementInterval で指定された値を解析するときに例外が発生した場合、Integration Server は両方の設定パラメータをデフォルト値にリセットします。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

Broker でトリガークライアントキューのポーリング頻度を制御する方法の詳細については、906 ページの「webMethods Messaging Triggerのポーリング要求の遅延」を参照してください。

watt.server.cors.allowedOrigins
Integration Server でクロスオリジン要求からリソースへのアクセスを許可する URI を指定します。

Integration Server は、CORS (クロスオリジンリソース共有) 標準の勧告に基づき、このパラメータの値を使用して要求の妥当性検査を行います。Integration Server では、CORS 要求の Origin ヘッダーで指定された URI をチェックします。ヘッダーの URI がこのパラメータで指定された URI と一致しない場合、Integration Server は要求を拒否し、次のメッセージと共に HTTP 403 状態コードを返します。

```
Specified Origin is not allowed
```

このパラメータで指定された値と URI が一致する場合、Integration Server は CORS Access-Control-Allow-Origin 応答ヘッダーを含む応答を要求元のユーザエージェントに返します。

このパラメータはカンマ区切りまたはスペース区切りの URI リスト、あるいはファイルへの絶対パスに設定できます。上記の組み合わせを指定することはできません。

カンマ区切りまたはスペース区切りの URI リストを使用してこのパラメータを定義するには、「*protocol://hostname*」または「*protocol://hostname:portnumber*」を入力します。*hostname* には、マシンの IP アドレスまたは名前を指定できます。値の大文字と小文字は区別されます。複数の値を指定する場合は、スペースまたはカンマ区切り文字を使用します。

テキストファイルを使用してこのパラメータを定義するには、値を次のように設定します。

`file:pathToFile\filename`

例: `watt.server.cors.allowedOrigins=file:c:\cors\allowedOriginsList.txt`

ファイルでは、watt.sever.cors.allowedOrigins に必要な構文と同じ構文を使用する必要があります。ただし、各行に許可されたオリジンサーバ URI または正規表現を記入する必要がある点が異なります。

URL http://*host*:*port*/invoke/wm.server.admin:refreshAllowedOrigins を呼び出すと、参照先のテキストファイルを Integration Server にロードする内部サービスを実行できます。

注:
IP アドレスとホスト名の使用は交換可能ではありません。ホスト名を指定して、対応する IP アドレスを使用するクロスオリジン要求が受信された場合 (またはその逆の場合)、Integration Server によって要求は拒否されます。

アスタリスク (*) を使用して、任意の URI またはオリジンを使用できるように指定できます。

注:
watt.server.cors.supportsCredentials が「`true`」に設定されている場合、アスタリスクを使用して URI を示すことはできません。

許可されたオリジンサーバのカンマ区切りリストで正規表現を使用することもでき、オリジンサーバのリストを保守しやすくなります。Integration Server は、「r:」で始まるカンマ区切りリストの値を正規表現として処理します。Integration Server は、「r:」で始まらない値を単純な文字列として処理します。サーバ設定パラメータでは、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html で説明されているように、Java の正規表現構文を使用します。正規表現の値が一致していると見なされるためには、正規表現の値が HTTP 要求の Origin ヘッダーの値全体と一致する必要があります。

例:

```
watt.server.cors.allowedOrigins=http://test1.domain.com,r:https?://.*.test2.domain.com:[0-9]+,r:.+\.[a-zA-Z]*-int.domain.com
```

Integration Server は最初の値「http://test1.domain.com」を単純な文字列として処理します。Origin ヘッダーにこの値がある場合、これは許可されます。

2 番目の値「r:https?://.*.test2.domain.com:[0-9]+」には正規表現が含まれています。「r:」は正規表現の一部ではありません。指定された Origin ヘッダーとの一致に使用される実際の正規表現は「https?://.*.test2.domain.com:[0-9]+」です。

3 番目の値「r:.+\.[a-zA-Z]*-int.domain.com」には正規表現が含まれています。「r:」は正規表現の一部ではありません。指定された Origin ヘッダーとの一致に使用される実際の正規表現は「.+\.[a-zA-Z]*-int.domain.com」です。

「Origin: http://test1.domain.com」は最初の値と等しいため、許可されます。

「Origin: http://my.test2.domain.com:8080」は 2 番目の値と一致するため、許可されます。

「Origin: https://my.test2.domain.com:8088」は 2 番目の値と一致するため、許可されます。

「Origin: http://my.test2.domain.com」は許可されません。ポート番号があったならば、2 番目の値に一致しています。

「Origin: nbps://example.prod-int.domain.com」は 3 番目の値と一致するため、許可されます。

「Origin: example.qa.staging-int.domain.com」は 3 番目の値と一致するため、許可されます。

「Origin: example.dev1-int.domain.com」は許可されません。ホスト名の 2 番目のトークンに数字が含まれていなければ、3 番目の値に一致していました。

ホスト名、IP アドレス、およびポートに一致する正規表現 (「r:.+」、「r:.*」など) には「*」と同じ効果があります。

注:
CORS が有効になっている場合、Integration Server はすべての要求に対して watt.server.cors.allowedOrigins の正規表現のリストを順次に評価します。Integration Server は、一致が見つかるか、またはリストのすべての正規表現が評価されるまで、各正規表現の正規表現一致操作を実行します。Software AG は、頻繁に一致する正規表現をカンマ区切りリストの先頭に置くことをお勧めします。

注:
watt.server.cors.enabled パラメータを「`true`」に設定した場合は、値を指定する必要があります。

watt.server.cors.enabled
Integration Server が CORS をサポートするかどうかを指定します。このパラメータを「`true`」に設定した場合は、Integration Server でクロスオリジン要求が有効になります。デフォルト値は「`false`」です。

注:
webMethods Enterprise Gateway 設定では、Enterprise Gateway Server でのみ CORS のサポートを有効にしてください。

Integration Server で CORS を使用する場合の詳細については、154 ページの「Integration Server での CORS の使用」を参照してください。クロスオリジン要求を許可するように Integration Server を設定する手順については、154 ページの「Integration Server の CORS 要求の受け入れの設定」を参照してください。

watt.server.cors.exposedHeaders
CORS 要求への応答で、Integration Server が CORS Access-Control-Expose-Headers ヘッダーに設定できる値を指定します。CORS Access-Control-Expose-Headers ヘッダーには、ユーザエージェントに公開できるヘッダーを定義します。クライアントサイドコードを確認し、応答ヘッダーがある場合は、どの応答ヘッダーがクライアントによって取得され、公開する必要があるのかを判別します。

このパラメータの値の大文字と小文字は区別されません。複数の値を指定する場合は、スペースまたはカンマ区切り文字を使用します。次に例を示します。

```
watt.server.cors.exposedHeaders=<value1,value2,value3>
```

CORS (Cross-Origin Resource Sharing) 標準の決定に従い、Cache-Control、Content-Language、Content-Type、Expires、Last-Modified、Pragma はシンプルな応答ヘッダーであると考えられ、常に公開されます。

watt.server.cors.host
クライアントが Integration Server にクロスオリジン要求を送信する場合のホストおよびポートを指定します。Integration Server は、この値を使用して、クロスオリジン要求の Host ヘッダーの妥当性検査を行います。Software AG では、セキュリティを向上させるため、CORS のサポートを有効にしている (つまり、watt.server.cors.enabled を「`true`」に設定している) 場合は、このパラメータを定義することをお勧めします。

このパラメータを定義するには、*hostname*:*port* または *hostname* を設定します。*hostname* には、マシンの IP アドレスまたは名前を指定できます。このパラメータの値の大文字と小文字は区別されます。このパラメータの値を指定しなかった場合、Integration Server は要求の Host ヘッダーの妥当性検査を行いません。値が指定されていても、要求の Host ヘッダーに一致しない場合、Integration Server はメッセージと共に 403 応答を返します。

注:
IP アドレスとホスト名の使用は交換可能ではありません。ホスト名を指定して、対応する IP アドレスを使用するクロスオリジン要求が受信された場合 (またはその逆の場合)、Integration Server によって要求は拒否されます。

watt.server.cors.maxAge
ユーザエージェントがプリフライト要求の結果をキャッシュできる時間を秒単位で指定します。Integration Server は、CORS Access-Control-Max-Age 応答ヘッダーにあるこの値を使用します。

デフォルトは `-1` で、これはクライアントが結果を保持しないことを示します。値には整数を指定する必要があります。

watt.server.cors.supportsCredentials
すべての CORS 要求への応答で、Integration Server が CORS Access-Control-Allow-Credentials ヘッダーを設定するかどうかを指定します。

「`true`」に設定した場合、Integration Server は応答の CORS Access-Control-Allow-Credentials ヘッダーを「true」に設定します。このパラメータを「true」に設定する場合は、以下の点に留意してください。

- ユーザのクレデンシャルが有効であり、リソースにアクセスする権限がユーザにある場合、Integration Server によってクレデンシャルによる CORS 単純要求が実行されます。ユーザエージェントによって、ユーザへの応答が表示されます。
- Integration Server が CORS プリフライト要求に応答する場合、その応答には、「`true`」に設定された Access-Control-Allow-Credentials ヘッダーが含まれるため、ユーザエージェントによって、クレデンシャルを含む要求が許可されます。

注:
ユーザのクレデンシャルには、Cookie、HTTP 認証またはクライアントサイド SSL 認証を指定できます。

「`false`」 (デフォルト) に設定した場合、Integration Server は応答の CORS Access-Control-Allow-Credentials ヘッダーを設定しません。このパラメータを「false」に設定する場合は、以下の点に留意してください。

- ユーザのクレデンシャルが有効であり、リソースにアクセスする権限がユーザにある場合、Integration Server によってクレデンシャルによる CORS 単純要求が実行されます。ユーザエージェントには、ユーザへの応答が表示されません。
- Integration Server が CORS プリフライト要求に応答する場合、その応答には Access-Control-Allow-Credentials ヘッダーが含まれないため、クレデンシャルを含む要求は、ユーザエージェントによって許可されません。

注:

Software AG では、セキュリティ上の理由から、このパラメータのデフォルト設定 (「false」) を使用することをお勧めします。「false」に設定した場合、ユーザエージェントは応答で送信された Cookie を無視します。このパラメータを「true」に設定した場合、Software AG ではリソースを保護するために CSRF ガードの使用を強くお勧めします。安全性を高めるには、許可されたリソースへのアクセスに、クレデンシャル付きの要求を使用するのではなく、OAuth など別の標準を使用することを検討してください。

watt.server.cors.supportedHeaders

Integration Server がクロスオリジン要求で許可する要求ヘッダーを指定します。Integration Server は、この値を使用して CORS Access-Control-Request-Header 要求ヘッダーの妥当性検査を行います。watt.server.cors.supportedHeaders で指定された値とヘッダーが完全に一致しない場合、Integration Server は要求を拒否し、次のメッセージと共に HTTP 403 状態コードを返します。

```
Header not supported
```

このパラメータで指定された値とヘッダーが一致する場合、Integration Server はこのパラメータの値が設定された CORS ヘッダー Access-Control-Allow-Headers を使用して応答します。

サーバサイドコードを確認し、応答ヘッダーがある場合は、どの応答ヘッダーがサーバアプリケーションによって読み込まれ、明示的に許可される必要があるのかを判断します。

このパラメータの値の大文字と小文字は区別されます。複数の要求ヘッダーを区切る場合はスペースまたはカンマを使用します。たとえば、次のように入力します。

```
watt.server.cors.supportedHeaders=<header1,header2,header3>
```

CORS (Cross-Origin Resource Sharing) 標準の決定に従い、Accept、Accept-Language、Content-Type (値が application/x-www-form-urlencoded の場合、「multiplart/form-data、または text/plain) は、シンプルな要求ヘッダーであると考えられ、常に公開されます。

watt.server.cors.supportedMethods

Integration Server がクロスオリジン要求で許可する HTTP メソッドを指定します。Integration Server は、この値を使用して CORS Access-Control-Request-Method 要求ヘッダーの値の妥当性検査を行います。watt.server.cors.supportedMethods で指定された値とヘッダーが完全に一致しない場合、Integration Server は要求を拒否し、次のメッセージと共に HTTP 403 状態コードを返します。

```
Method not supported
```

このパラメータで指定された値とヘッダーが一致する場合、Integration Server はこのパラメータの値が設定された CORS ヘッダー Access-Control-Allow-Headers を使用して応答します。

複数の HTTP メソッドを区切る場合はスペースまたはカンマを使用します。指定可能な値は、OPTIONS、HEAD、GET、POST、PUT、PATCH および DELETE です。Integration Server は、デフォルトでこれらのいずれの値も受け入れます。

watt.server.createPackage.ignorePattern

Integration Server でパッケージをパブリッシュするときに除外するファイルタイプのセミコロン区切りのリストを指定します。このパラメータを使用して、複製によるパッケージのパブリッシュ時とパッケージの展開時に除外するファイルタイプを指定できます。たとえば、Integration Server で .svn ファイルをパブリッシュしないようにするには、パラメータを watt.server.createPackage.ignorePattern=.svn のように指定します。

複数のファイルタイプを指定するには、次のように区切り文字としてセミコロン (;) を使用します。

```
watt.server.createPackage.ignorePattern=.svn;.java;.xml
```

watt.server.cronMaxThreads
Integration Server でスケジュール済みシステムタスクに使用される cronjob ベースのスレッドプール用に保持される最大スレッド数です。この最大スレッド数に達した場合、Integration Serverは、現在のプロセスが完了してスレッドがプールに戻るまで待機してから、次のプロセスを実行します。デフォルトは 5 です。

watt.server.cronMinThreads
Integration Server でスケジュール済みシステムタスクに使用される cronjob ベースのスレッドプール用に保持される最小スレッド数です。Integration Server 起動時の初期のスレッドプールにはこの最小スレッド数が含まれます。デフォルトは 2 です。

watt.server.dashboard.api.responseTime.threshold
API の応答 (実行) 時間が、Dashboard の [API] タブで、応答時間の [**最大値**] 列の背景色を赤にして警告するほど遅いかどうかを決定するときに使用するしきい値を指定します。

しきい値は、[API] タブで赤い背景色を使用する根拠となる、サービスの特定の実行が遅すぎるかどうかを決定するために使用される、既存の平均 API 応答時間に掛けるパーセンテージです。

ゼロより大きく、符号なしの整数データタイプの最大値である 4294967295 よりも小さな整数を指定します。デフォルトは 50 です。

要件に準拠していない値が指定された場合は、Integration Server がパラメータをデフォルトの 50 にリセットします。

Integration Server は、特定の API 実行の最大応答時間に警告が必要かどうかを決定するための次の式でしきい値を使用します。

最大応答時間 > 平均応答時間 + (平均応答時間 x しきい値%)

ここで、

平均応答時間は、Dashboard の右上にある選択された時間間隔内のすべての API の呼び出しの応答時間の平均です。

しきい値は watt.server.dashboard.api.responseTime.threshold の値です。

例えば、watt.server.dashboard.api.responseTime.threshold が 20 に設定してあり、myRestAPI の平均応答時間が 100 ミリ秒であるとします。

100 ms + (100 ms x 0.20) = 120 ms

myRestAPI の呼び出しの最大応答時間が 120 ミリ秒を超えた場合、Dashboard の [API] タブにある myRestAPI の応答時間の最大値のセルの背景色が赤になります。

watt.server.dashboard.diskspace.threshold
ダッシュボードの [概要] タブで、ディスクスペースグラフの赤い部分を決定するために使用するしきい値を総ディスクスペースに対するパーセンテージで指定します。0 から 100 の間の整数を指定します。デフォルト値は 20 です。

このパラメータは、Microservices ライセンスを使用していない Integration Server に使用されます。Microservices Runtime または Microservices ライセンスを使用している Integration Server では、ディスクスペースヘルスインジケータの [**空きディスク容量のしきい値**] プロパティの値がディスクスペースグラフの赤い部分を計算するために使用されます。

watt.server.dashboard.memory.threshold
ダッシュボードの [概要] タブで、メモリグラフの赤い部分を決定するために使用するしきい値を総メモリに対するパーセンテージで指定します。0 から 100 の間の整数を指定します。Integration Server のデフォルト値は 20 ミリ秒です。

このパラメータは、Microservices ライセンスを使用していない Integration Server にのみ使用されます。Microservices Runtime または Microservices ライセンスを使用している Integration Server では、メモリヘルスインジケータの [**空きメモリのしきい値**] プロパティの値がメモリグラフの赤い部分を計算するために使用されます。

watt.server.dashboard.service.responseTime.threshold
特定のサービスの応答 (実行) 時間が、ダッシュボードの [サービスインスタンス] タブで応答時間の [最大値] 列の背景色を赤にして警告するほど遅いかどうかを決定するときに使用するしきい値を指定します。

しきい値は、[サービスインスタンス] タブで赤い背景色を使用する根拠となる、サービスの特定の実行が遅すぎるかどうかを決定するために使用される、既存の平均サービス応答時間に掛けるパーセンテージです。

ゼロより大きく、符号なしの整数データタイプの最大値である 4294967295 よりも小さな整数を指定します。デフォルトは 50 です。

要件に準拠していない値が指定された場合は、Integration Server がパラメータをデフォルトの 50 にリセットします。

Integration Server は、特定のサービス実行の最大応答時間に警告が必要かどうかを決定するための次の式でしきい値を使用します。

最大応答時間 > 平均応答時間 + (平均応答時間 x しきい値%)

ここで、

平均応答時間は、Dashboard の右上にある選択された時間間隔内のすべてのサービス呼び出しの応答時間の平均です。

しきい値は `watt.server.dashboard.service.responseTime.threshold` の値です。

たとえば、`watt.server.dashboard.service.responseTime.threshold` が 20 に設定してあり、serviceA の平均応答時間が 100 ミリ秒であるとします。

100 ms + (100 ms x 0.20) = 120 ms

serviceA の呼び出しの最大応答時間が 120 ミリ秒を超えた場合、ダッシュボードの [サービスインスタンス] タブにある serviceA の応答時間の最大値のセルの背景色が赤になります。

watt.server.dateStampFmt
ログファイルに使用する日付形式を指定します。サーバログの場合は、このパラメータにより、サーバログファイルおよび Integration Server Administrator で日付形式を表示する方法を指定します。サービス、エラー、セッション、保証付きデリバー、セキュリティのログなど、その他のすべてのログの場合は、このパラメータにより、Integration Server Administrator で日付形式を表示する方法のみを指定します。

注:
このパラメータを使用して、サービス、エラー、セッション、保証付きデリバー、セキュリティなどのログファイルの日付形式を指定するには、watt.server.audit.displayLogs.convertTime を「`true`」に設定する必要があります。使用方法については、watt.server.audit.displayLogs.convertTime を参照してください。

使用する日付形式の指定には、Java クラス `java.text.SimpleDateFormat` でサポートされている任意の形式を使用できます。たとえば、08-12-02 14:44:33:1235 という形式で日付を表示するには、「dd-MM-yy HH:mm:ss:SSSS」と指定します。

watt.server.date.SuppressPatternError
pub.date:dateTimeFormat サービスに入力値が渡されない場合のサーバの応答動作を指定します。「`true`」に設定した場合は、*value* パラメータに対して単に NULL 値が戻されます。デフォルトでは、サーバは例外をスローします。

watt.server.db.blocktimeout
データベースへの接続を待っているときに、サーバが要求をブロックする最大時間をミリ秒で指定します(データベースは、WmDB パッケージ内のエイリアスで指定する必要があります)。デフォルトでは無期限に待機するよう設定されています。また「-1」を指定した場合にも、無期限に待機します。このプロパティは、すべてのプールに対してグローバルに適用されます。

watt.server.db.connectionCache
サーバがデータベースへの接続を管理する方法を指定します。

「`server`」と指定すると、エイリアスによってサーバに定義された各データベース用の接続のプールを管理するようにサーバに指示します。 プールが最大接続数に達しているために要求を満たすことができない場合、サーバは要求を実行せず、後で再度試行します。

「`session`」と指定すると、セッションごとにデータベース接続が確立されます。つまり、サーバは、データベース接続を必要とするサービス要求を受信したときに、そのセッションの接続が確立されていない場合は新しい接続を作成し、そうでない場合は前に作成されたそのセッションの接続を使用します。接続に使用可能なスロットがデータベースにないなどの理由で、セッションの接続を確立できないと、要求は失敗します。デフォルトは「session」です。

データベース接続プーリングを有効にすると、サーバに定義された各データベースに対して自動的にプールが作成されますが、Integration Server Administrator の [**エイリアス情報の編集**] ページで各プールの特性を個別に制御できます。データベースへの接続設定の詳細については、『*WmDB User's Guide*』を参照してください。

watt.server.db.maintainminimum
タイムアウトになった非アクティブ接続の消去をサーバが処理する方法を指定します。このパラメータを「`false`」に設定すると、サーバは非アクティブ接続をすべて消去します。このパラメータが「`true`」に設定されていると、サーバは非アクティブ接続を消去しますが、プール内の接続数が最小値に達すると、消去を停止します。エイリアスを定義するときに、この最小値を指定します。このプロパティは、すべてのプールに対してグローバルに適用されます。

watt.server.db.share.ISInternal
Integration Server がクラスタ化されていない場合にも、Integration Server Administrator の [JDBC プール] ページで、Integration Server の [ISInternal 機能エイリアス] が、他の Integration Server で使用しているデータベースと同じデータベースを参照するかどうかを指定します。Integration Server が同じ ISInternal データベースを共有するよう指定するには、このプロパティを「`true`」に設定します。クラスタ化はされていないが ISInternal データベースは共有する、複数の Integration Server で保証付きデリバーを使用する場合は、このプロパティを「`true`」に設定する必要があります。このプロパティは、ISInternal データベースを共有するすべての Integration Server について「`true`」に設定する必要があります。Integration Server

が他の Integration Server と ISInternal データベースを共有しない場合は、このプロパティを「false」に設定します。デフォルトは「false」です。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.db.testSQL
JDBC 接続プールでデータベース接続のテストに使用できる SQL ステートメントを指定します。呼び出し側が JDBC 接続プールから JDBC 接続を要求する場合に watt.server.db.testSQL パラメータで SQL ステートメントが指定されている場合は、Integration Server は JDBC 接続プールで使用されているデータベースに対して指定された SQL ステートメントを実行します。SQL ステートメントの実行が成功すると、Integration Server はデータベース接続が有効であると判断して、呼び出し側に接続を返します。SQL ステートメントの実行に成功しない場合は、Integration Server はデータベース接続が無効であると判断して、JDBC 接続プールから削除します。Integration Server は新しい JDBC 接続を作成し、JDBC 接続プールに配置し、呼び出し側に新しい接続を返します。watt.server.db.testSQL で値が指定されない場合に呼び出し側が JDBC 接続を要求すると、Integration Server は接続プール内のデータベース接続をテストしません。パラメータのデフォルト値はありません。

注:
JDBC 接続が要求されるたびにデータベース接続をテストすると、テストのたびにデータベースに対して SQL ステートメントが実行されるので、パフォーマンスに影響する可能性があります。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.debugFIPS
ユーザが認証を受けるたびに Integration Server が FIPS 暗号化プロバイダをチェックするかどうかを指定します。このパラメータを「true」に設定した場合、ユーザが認証を受けるたびに、Integration Server は FIPS 暗号化プロバイダをチェックするために、このプロバイダを使用して java.security.MessageDigest インスタンスを作成します。プロバイダのチェックでは、プロバイダの名前および生成された MessageDigest を標準エラーに書き込みます。「false」に設定した場合、Integration Server はユーザが認証を受けるたびに暗号化プロバイダをチェックしません。デフォルトは「false」です。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.defaultContentHandler
Integration Server が text/html コンテンツタイプを登録するコンテンツハンドラを指定します。「true」 (デフォルト) に設定すると、Integration Server は text/html コンテンツタイプを ContentHandler_Default コンテンツハンドラに登録します。「false」に設定すると、Integration Server は text/html コンテンツタイプを ContentHandler_CGI コンテンツハンドラに登録します。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.defaultCountry

ResourceBundles から文字列を選択してクライアントに返すときに Integration Server が使用するデフォルトの国 (Java ロケールのコンポーネント) を指定します。すべての ISO 国コードを有効な値として指定できます。このプロパティのデフォルト値は「us」です。デフォルトのロケールは「en-US」です。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.defaultLanguage
ResourceBundles から文字列を選択してクライアントに返すときに Integration Server が使用するデフォルトの言語 (Java ロケールのコンポーネント)。すべての ISO 言語コードを有効な値として指定できます。このプロパティのデフォルト値は「en」です。デフォルトのロケールは「en-US」です。

重要:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.defaultPackage
デフォルトのパッケージの名前を指定します。この値が設定されていない場合、Integration Server は WmAdmin をデフォルトのパッケージとして使用します。Integration Server をセーフモードで起動すると、WmAdmin はこのパラメータの値に関係なくデフォルト値になります。

注:
このパラメータの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.deprecatedExceptionLogging
Integration Server でサービスの例外を Integration Server エラーログに記録する方法を指定します。基本的な例外ログを記録する場合はこのパラメータを「true」に設定し、例外の原因の特定に役立つ詳細な例外ログを記録する場合は「false」 (デフォルト設定) に設定します。

設定値	Integration Server
true	■ ネストされたスタックに各サービスの例外が積み上げられると、そのスタックのサービスの例外を記録します。親サービスが例外をキャッチしても、それを再スローしない場合、Integration Server はスタックに積み上げられた各子サービスの例外を引き続き記録します。 ■ 各サービスのスタックトレースを表示します。
false	■ ネストされたサービスで例外が発生しても、最上位サービスの例外を 1 回だけ記録します。スタックのサービスが例外をキャッチしても、それを再スローしない場合、Integration Server は例外を記録しません。 **注:** 各コールスタックのサービス、またはサービスで使用されるコアクラスのいずれかで例外が明示的に記録される場合は、例外が複数回記録されることがあります。

設定値	Integration Server

- 記録された例外に関して、例外が最初に発生した最深部のサービスのスタックトレースを表示します。watt.server.deprecatedExceptionLogging を「true」に設定すると、多くの場合、このスタックトレースはログから切り捨てられます。
- ネストされた各例外からのエラーメッセージを連結します。

このパラメータを「true」に設定すると、例外の原因を追跡することが難しくなります。このため、Software AG では、例外をキャッチしても再スローしないサービスを実行中でない限り、このパラメータは「true」に設定しないことをお勧めします。

エラーログの解釈方法およびデバッグサービスにログを役立てる方法の詳細については、1003 ページの「エラーログによるサービス例外のデバッグ」を参照してください。

watt.server.diagnostic.logFiles.maxMB
Integration Server が診断データの収集中に、監査ログのためにファイルシステムから読み取るデータの最大メガバイト数を指定します。このパラメータは監査ログファイルの収集にのみ影響を与えます。データベースから読み取られる監査ログファイルは影響を受けず、サーバログ、stats ログ、terracotta-client ログなど、他のログファイルも影響を受けません。デフォルトは 250 メガバイトです。このパラメータの変更を適用するために Integration Server を再起動する必要はありません。

watt.server.diagnostic.logperiod
診断ツールを実行したときに、何時間分のログが返されるかを指定します。デフォルトは 6 です。このプロパティを 0 (ゼロ) に設定すると、診断ユーティリティはログファイルを返さず、設定ファイルおよびランタイムデータファイルのみを返します。

watt.server.diagnostic.port
Integration Server が応答しないときにアクセスするために使用する診断ポートを指定します。インストール中に、Integration Server は診断ポートを自動的に 9999 に作成します。デフォルトは 9999 です。

> **注:**
> この設定は、[診断ポート設定の編集] ページの [**ポート**] 設定よりも優先します。

同じホストマシン上で複数の Integration Server を実行している場合、各サーバの診断ポートにそれぞれ固有のポート番号を設定する必要があります。各 Integration Server インスタンスについて、一意のポート番号を使用するようにこのパラメータを設定します。診断ポートについて、および複数の Integration Server インスタンスを実行する方法の詳細については、211 ページの「HTTP 診断ポートの追加」を参照してください。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。Integration Server を再起動せずに診断ポートを変更するには、Integration Server Administrator の **[サーバ] > [ポート]** でデフォルトの診断ポートを編集します。詳細については、228 ページの「ポートの編集」を参照してください。

watt.server.diagnostic.tabular

Integration Serverが診断ファイルを表形式で生成するかどうかについて指定します。「false」に設定すると、Integration Server はファイルを表以外の形式で生成します。デフォルトは「true」です。

watt.server.dispatcher.comms.brokerPing
トリガー (Broker クライアント) が Broker に ping を実行する頻度 (ミリ秒単位) を指定します。Integration Server と Broker との間にファイアウォールがあると、ファイアウォールは接続がアイドル状態になったときにトリガーと Broker との接続を切断します。接続がアイドル状態にならないようにするため、トリガー Broker クライアントは webMethods Broker に定期的に ping を実行します。デフォルトは 1800000 ミリ秒 (30 分) です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.dispatcher.comms.connectionShareLimit
Integration Server が Broker と通信するために使用できる BrokerClient オブジェクトの数を指定します。Integration Server は、この値を渡して、次の Broker API を設定します。
COM.activesw.api.client.BrokerConnectionDescriptor.setConnectionShareLimit (int value)
詳細については、*webMethods Broker Java Client API Reference*を参照してください。
各 Broker 接続は、Broker に対して開いているソケットを表します。プロパティの値が小さいほどスループットは増加しますが、同時に、Integration Server が使用するリソースとファイル記述子の数も増加します。このパラメータの値を「0」または 1 より大きい値に設定してください。「0」に設定した場合、接続共有が無効になり、各 Broker クライアントは自身の Broker 接続を使用します。デフォルトは 5 です。

注:
Broker では「1」の値をサポートしていません。「1」を設定すると、Broker は例外を発行します。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.dispatcher.join.reaperDelay
Integration Server が完了した結合と期限切れの結合の状態情報を削除する頻度 (ミリ秒単位) を指定します。デフォルトは 1800000 ミリ秒 (30 分) です。

watt.server.dispatcher.messageStore.redeliverOriginalMessage
webMethods messaging triggerによって呼び出されたトリガーサービスの再試行失敗が発生した場合に、Integration Server で後続のトリガーサービス実行に関して元のメッセージと変更後のメッセージのどちらを格納および再デリバーするかを指定します。「true」に設定した場合、Integration Server は webMethods messaging triggerに送信された元のメッセージを格納および再デリバーします。「false」に設定した場合、Integration Server は再試行失敗が発生した時点のメッセージを格納します。Integration Server は後続の処理に関して変更後のメッセージを webMethods messaging triggerにデリバーします。デフォルトは「false」です。

watt.server.displayDirectories
ブラウザのユーザが Integration Server Administrator を使用しなくても Integration Server にあるディレクトリを表示できるようにするかどうかを指定します。このパラメータを「true」(デフォルト) に設定した場合、ユーザは Integration Server のディレクトリを表示できます。このパラメータを「false」に設定した場合、ディレクトリは表示されません。

watt.server.dumpWhitelist
ホワイトリストクラスフィルタ処理を有効にした場合、Integration Server で、デシリアライズを試行したが whiteslistclasses.xml ファイルの一部でないクラスを示すログファイルを生成するかどうかを示します。watt.server.dumpWhitelist が「true」に設定されている場合、Integration Server では、呼び出されたがホワイトリストクラスファイル内になかったクラスを示すログファイルを生成します。具体的には、Integration Server はシャットダウン時に次のファイルを作成します。

Integration Server_directory /instances/*instanceName*/logs/runtimeClasslist.*time*.xml。*time* は 1970 年 1 月 1 日を起点とする時間です (ミリ秒単位)。

watt.server.dumpWhitelist のデフォルト値は「false」です。これは、Integration Server では、呼び出されたがホワイトリストクラスファイルの一部でなかったクラスのログを生成しないことを示します。

watt.server.email.charset
電子メールメッセージ内の ASCII 以外の文字を含む部分をエンコードするために使用されるデフォルトの文字セットを指定します。ASCII 以外の文字を含む電子メールフィールドは、件名、MIME ヘッダー、本文テキストおよび添付ファイルです。このプロパティは、pub.client:smtp サービスで使用されます。デフォルトは「`utf-8`」です。pub.client:smtp サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.email.from
エラーに関する電子メールを送信するときに、「送信者」のアドレスとしてサーバが提示する電子メールアドレスを指定します。デフォルトでは、IntegrationServer@*localhost* が「送信者」のアドレスとして使用されます。この場合、*localhost* は Integration Server が実行されているホストの名前です。

watt.server.email.processReplyEmails
Integration Server が電子メールポートで受信する返信電子メールで件名行にプリフィックス「Re:」が含まれている場合に、その電子メールを処理するかどうかを指定します。件名行にプリフィックス「Re:」および有効なサービス名が含まれている返信電子メールを Integration Server に処理させる場合は、watt.server.email.processReplyEmails プロパティを「`true`」に設定します。このプロパティが「`false`」に設定されている場合、Integration Server は件名行にプリフィックス「Re:」が含まれていても電子メールを処理しません。代わりに、Integration Server はサーバログメッセージを生成します。デフォルトは「`false`」です。

watt.server.email.waitForServiceCompletion
Integration Server でメッセージ抽出用のスレッドをスレッドプールに戻す前に、電子メールポートがメッセージの処理を完了するまで待機するかどうかを指定します。

電子メールポートで受信したメッセージを抽出および処理するとき、Integration Server では、メッセージの抽出用として 1 個のスレッドを使用し、メッセージを処理するサービスの実行用として別のスレッドを使用します。デフォルトでは、スレッドを使用してメッセージ

を抽出したら、Integration Server はそのスレッドをサーバスレッドプールに戻し、他の処理の実行用として使用できるようにします。

電子メールポートで短期間に多数のメッセージを受信した場合、メッセージの抽出および処理のためにサーバスレッドプールが独占されることがあります。また、メッセージを処理するサービスの動作速度が遅いと、電子メールポートでのメッセージの処理が Integration Server のパフォーマンスに悪影響を及ぼす可能性があります。

この問題を避けるために、watt.server.email.waitForServiceCompletion プロパティを設定して、電子メールポートで受信したメッセージの処理に使用されるスレッド数を制限できます。

このプロパティを「`true`」に設定した場合、Integration Server は、メッセージを処理するサービスが完了するまで、メッセージの抽出に使用したスレッドをスレッドプールに戻しません。そのため、電子メールポートのメッセージの抽出および処理に使用される最大スレッド数は、2、または [**スレッドの数 (マルチスレッド処理がオンになっている場合)**] フィールドの値の 2 倍になります。

- 電子メールポートが POP3 電子メールサーバ用の場合、Integration Server では最大で 2 個のスレッドを使用して、ポートのメッセージを抽出および処理できます (電子メールポートで受信したメッセージの抽出用として 1 個のスレッドを使用し、メッセージを処理するサービスの実行用として 1 個のスレッドを使用します)。
- 電子メールポートが IMAP 電子メールサーバ用であり、マルチスレッド処理が無効になっている場合、Integration Server では最大で 2 個のスレッドを使用して、ポートのメッセージを抽出および処理できます (電子メールポートで受信したメッセージの抽出用として 1 個のスレッドを使用し、メッセージを処理するサービスの実行用として 1 個のスレッドを使用します)。
- 電子メールポートが IMAP 電子メールサーバ用であり、マルチスレッド処理が有効になっている場合、Integration Server では [**スレッドの数 (マルチスレッド処理がオンになっている場合)**] フィールドの値の 2 倍の数のスレッドを使用して、ポートのメッセージを抽出および処理できます。たとえば、[**スレッドの数 (マルチスレッド処理がオンになっている場合)**] フィールドの値が `5` の場合、Integration Server では最大で 10 個のスレッドを使用して、電子メールポートのメッセージを抽出および処理できます (5 個のスレッドを使用して電子メールポートのメッセージを同時に抽出し、さらに 5 個のスレッドを使用してそれらのメッセージを処理します)。

watt.server.email.waitForServiceCompletion を「`false`」に設定した場合、Integration Server は、メッセージを処理するサービスが完了するのを待たずに、メッセージの抽出に使用したスレッドをスレッドプールに戻します。

デフォルトは「`false`」です。

> **注:**
> watt.server.email.waitForServiceCompletion パラメータは、Integration Server 上のすべての電子メールポートに適用されます。

watt.server.enableHotDeployment

Integration Server でパッケージのホット展開が有効かどうかを指定します。「true」に設定すると、ホット展開は有効化されます。「false」に設定すると、ホット展開は無効化されます。デフォルトは「false」です。

watt.server.enableHotDeployment の値は、Integration Server Administrator の [**設定**] > [ホット展開] > [設定の編集] ページの [**有効**] フィールドに関連付けられています。

watt.server.enterprisegateway.ignoreXForwardedForHeader
Integration Server が、Enterprise Gateway Server 内でルールが処理されているときに、X-Forwarded-For 要求ヘッダーを無視するかどうかを指定します。このプロパティを「true」に設定すると、Integration Server は X-Forwarded-For 要求ヘッダーを無視し、プロキシサーバの IP アドレスをホスト IP アドレスとみなします。このプロパティを「false」に設定すると、Integration Server は、実際のホスト IP アドレスを X-Forwarded-For 要求ヘッダーから取得します。デフォルト値は「true」です。

watt.server.errorMail
Integration Server から重大なサーバログエントリに関するメッセージを送信する電子メールアドレスを指定します。デフォルトはありません。

watt.server.event.audit.async
監査イベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、監査イベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、監査イベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.audit.async パラメータの値は無視されます。

watt.server.event.exception.async
例外イベント、エラーイベントまたはジャーナルイベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、例外イベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、例外イベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.exception.async パラメータの値は無視されます。

watt.server.event.gd.async
保証付きデリバーイベント (gdStart および gdEnd) のイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、保証付きデリバーイベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、保証付きデリバーイベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.gd.async パラメータの値は無視されます。

watt.server.event.jmsDeliveryError.async
JMS デリバー失敗イベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、JMS デリバー失敗イベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、JMS デリバー失敗イベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.jmsDeliveryError.async パラメータの値は無視されます。

watt.server.event.jmsRetrievalError.async
JMS 抽出失敗イベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、JMS 抽出失敗イベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、JMS 抽出失敗イベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.jmsRetrievalError.async パラメータの値は無視されます。

watt.server.event.replication.async
複製イベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、複製イベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、複製イベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.replication.async パラメータの値は無視されます。

watt.server.event.routing.bodyAsString
pub.event.routing:subscribe サービスを使用している場合、サブスクリプションサービスの入力として evt:Event ドキュメントを受信します。このパラメータは、evt:Event ドキュメント内の evt:body 要素を、想定されるドキュメントタイプにマッピングするかどうかを指定します。このパラメータを「false」に設定した場合、Integration Server は evt:Event ドキュメント内の evt:body 要素を、想定されるドキュメントタイプにマッピングします。このパラメータを「true」に設定した場合、Integration Server は evt:Event ドキュメント内の evt:body 要素をマッピングしません。デフォルトは「false」です。

watt.server.event.routing.runAsUser

pub.event.routing:send および pub.event.routing:subscribe 組み込みサービスで指定されたサービスを呼び出すユーザを指定します。pub.event.routing:send および pub.event.routing:subscribe サービスの *runAsUser* 入力パラメータでユーザが指定されていない場合に限り、Integration Server は、このパラメータで指定されたユーザを使用します。デフォルトは Administrator です。

このユーザは、内部的にも外部ディレクトリでも定義することができますが、指定されたサービスを呼び出すのに適切な ACL 権限を持つ既存のグループに属する必要があります。Integration Server は、サービスが実行されるまで、このユーザの妥当性検査を行いません。

このパラメータの新しい値は、既存のサブスクリプションと同様に pub.event.routing:subscribe を使用して作成される新しいサブスクリプションにも適用されます

watt.server.event.security.async
セキュリティイベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「`true`」に設定した場合、Integration Server は、セキュリティイベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「`false`」に設定した場合、Integration Server は、セキュリティイベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「`true`」です。

注:
スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.security.async パラメータの値は無視されます。

watt.server.event.session.async
セッションイベント (sessionStart、sessionEnd および sessionExpire) のイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「`true`」に設定した場合、Integration Server は、セッションイベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「`false`」に設定した場合、Integration Server は、セッションイベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「`true`」です。

注:
スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.session.async パラメータの値は無視されます。

watt.server.event.stat.async
統計イベントのイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「`true`」に設定した場合、Integration Server は、統計イベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「`false`」に設定した場合、Integration Server は、統計イベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「`true`」です。

注:
スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.stat.async パラメータの値は無視されます。

watt.server.event.tx.async
トランザクションイベント (txStart および txEnd) のイベントハンドラを非同期的に呼び出すか、または同期的に呼び出すかを指定します。このパラメータを「true」に設定した場合、Integration Server は、トランザクションイベントをサブスクライブしているイベントハンドラ (サービス) を非同期的に呼び出します。このパラメータを「false」に設定した場合、Integration Server は、トランザクションイベントをサブスクライブしているイベントハンドラを同期的に呼び出します。デフォルトは「true」です。

> **注:**
> スレッドプールが枯渇した場合、Integration Server はデッドロックを防ぐため、すべてのイベントを同期的に処理します。つまり、イベントストームの間、watt.server.event.tx.async パラメータの値は無視されます。

watt.server.eventHandlerCreateSession
Integration Server が Event Manager によって呼び出されるサービスのセッションを作成するかどうかを制御します。「true」に設定すると、Integration Server はセッションを作成し、呼び出されたサービスと関連付けます。「false」に設定すると、Integration Server は Event Manager によって呼び出されるサービスのセッションを作成しません。デフォルトは「false」です。

> **注:**
> 「true」に設定すると、watt.server.eventHandlerSessionTimeout パラメータを使用してセッションのタイムアウト値を指定します。

watt.server.eventHandlerSessionTimeout
watt.server.eventHandlerCreateSession が「true」に設定されている場合に、watt.server.eventHandlerSessionTimeout は、Event Manager によって呼び出されるサービス用に Integration Server によって作成されるセッションのタイムアウト値を指定します。デフォルト値は 60000 ミリ秒です。

watt.server.eventHandlerUser
Integration Server でイベントハンドラサービスの実行に使用するクレデンシャルを持つユーザアカウントを指定します。ローカルに定義されたユーザアカウント、またはセントラルディレクトリか外部ディレクトリに定義されたユーザアカウントを指定できます。指定したユーザアカウントには、イベントハンドラとして使用されているサービスに割り当てられた実行 ACL に必要なクレデンシャルが含まれていることを確認してください。デフォルトは Administrator です。

watt.server.eventMgr.maxThreads
Integration Server が Event Manager スレッドプールに使用するスレッドの最大数を指定します。デフォルトは 5 です。

Integration Server は、この値が watt.server.eventMgr.minThreads よりも大きいことを確認します。watt.server.eventMgr.maxThreads の値が watt.server.eventMgr.minThreads の値よりも小さい場合、Integration Server は watt.server.eventMgr.maxThreads の値を watt.server.eventMgr.minThreads より 1 大きく設定します。たとえば、watt.server.eventMgr.minThreads が 6、watt.server.eventMgr.maxThreads が 5 に設定されている場合、Integration Server は watt.server.eventMgr.maxThreads の値を 7 に設定します。

watt.server.eventMgr.minThreads

Integration Server が Event Manager スレッドプールに使用するスレッドの最小数を指定します。デフォルトは 2 です。

Integration Server は、この値が 1 よりも大きいことを確認します。1 より小さい値に設定すると、Integration Server によって 1 にリセットされます。

watt.server.fileEncoding
サーバがテキストファイルの読み書きに使用するエンコーディングを指定します。この設定は Unicode として保存されているファイルに影響を与えません。デフォルトは、お使いの JVM の file.encoding プロパティです。

> **注:**
> 受信 XML ファイルを処理するには、watt.server.xml.encoding パラメータを使用します。watt.server.fileEncoding パラメータで指定した文字エンコーディングを使用して受信 XML ファイルを処理するように Integration Server を設定した場合は、watt.server.fileEncoding パラメータの値が watt.server.xml.encoding に指定した値と同じものに設定されていることを確認する必要があります。

watt.server.ftpConnect.message
FTP クライアントが接続要求を発行したときに、Integration Server がそのクライアントに返す 220 メッセージの内容を指定します。

このパラメータを「`true`」 (デフォルト) に設定した場合、Integration Server は次のメッセージを FTP クライアントに返します。

```
Connected to IS_hostname.
220 IS_hostname FTP server (webMethods Integration Server version n.n.n.n) ready.
```

このパラメータを「`false`」に設定した場合、Integration Server は次のメッセージを FTP クライアントに返します。

```
Connected to IS_hostname.220
```

このパラメータを他の値に設定した場合、その値が 220 メッセージで返されます。たとえば、「`Custom FTP connect message`」と指定すると、Integration Server は次のメッセージを FTP クライアントに返します。

```
Connected to IS_hostname.220 Custom FTP connect message.
```

watt.server.ftp.listingFileAge
FTP サーバとして機能している Integration Server で更新または作成されたファイルにアクセスできるようになる前に経過する必要がある秒数を指定します。このパラメータで指定された時間内に作成または更新されたファイルは、FTP LIST コマンドの結果には含まれません。デフォルト値は 60 秒です。

watt.server.ftpLogin.message
FTP クライアントがログイン要求を発行したときに、Integration Server がそのクライアントに返す 230 メッセージの内容を指定します。

このパラメータを「`true`」 (デフォルト) に設定した場合、Integration Server は「`230 User userid logged in`」というメッセージを FTP クライアントに返します。

このパラメータを「`false`」に設定した場合、Integration Server は「`230`」というメッセージを FTP クライアントに返します。

このパラメータを他の値に設定した場合、その値が 230 メッセージで返されます。たとえば、「Custom FTP login message」と指定すると、Integration Server は次のメッセージを FTP クライアントに返します。

```
230 Custom FTP login message.
```

watt.server.ftpSystem.message

FTP クライアントがシステム要求を発行したときに、Integration Server がそのクライアントに返す 215 メッセージの内容を指定します。このパラメータを「true」 (デフォルト) に設定した場合、Integration Server は次のメッセージを FTP クライアントに返します。

```
215 UNIX Type: L8 Version: webMethods IS FTP <Integration Server n.n.n.n>
```

このパラメータを「false」に設定した場合、Integration Server は次のメッセージを FTP クライアントに返します。

```
215
```

このパラメータを他の値に設定した場合、その値が 215 メッセージで返されます。たとえば、「Custom FTP system message」と指定すると、Integration Server は次のメッセージを FTP クライアントに返します。

```
215 Custom FTP system message.
```

watt.server.ftp.usecommandip

FTP サーバへの接続の際に、pub.client:ftp サービスで NAT サーバからの接続情報が使用されるかどうかを制御します。

pub.client:ftp サービスが FTP サーバとの間のデータ転送を試みると、Integration Server は最初に、pub.client:ftp サービスで指定されている IP アドレスを持つ FTP サーバに接続します。FTP サーバはこれに応答して、転送を実行するために Integration Server が接続するべき FTP サーバの IP アドレスを返信します。FTP サーバが NAT サーバの内側にある場合、NAT サーバはこのアドレスを取得して変換した後、Integration Server にアドレスを送信します。

このプロパティは、Integration Server が、NAT サーバから提供されるアドレスを使用するか、pub.client:ftp サービスを通じてあらかじめ指定されているアドレスを使用するかを制御します。

このパラメータが「true」に設定されている場合、Integration Server はアドレス変換を省略して、サービスによって指定されているアドレスを参照します。これが失敗した場合、Integration Server は例外をスローします。

このパラメータがデフォルトの「false」に設定されている場合、Integration Server は NAT サーバから提供されたアドレスを参照します。これが失敗した場合、Integration Server は、pub.client:ftp サービスで指定されている IP アドレスを参照します。いずれの試みも失敗した場合、Integration Server は例外をスローします。

watt.server.grpc.clientCertRequired

gRPC サーバが認証にクライアント認証の使用を要求するか、またはクライアント認証を必須にするかを指定します。「true」に設定した場合、gRPC サーバはすべての gRPC クライアント要求に対してクライアント認証を必須にします。gRPC サーバは、クライアント認証を含まない要求を拒否します。「false」に設定した場合、gRPC サーバはすべての gRPC クライアント要求に対してクライアントクレデンシャルを要求します。ただし、認証が提供されない場合、Integration Server は AuthUser および AuthPass メタデータヘッダーでクライアント認証をチェックします。AuthUser および AuthPass が指定されていない場合、Integration Server

はクライアントクレデンシャルとしてデフォルトユーザを使用します。「true」の値の方が、安全性が高いオプションです。デフォルトは「false」です。

`watt.server.grpc.enable.TLS` を「true」に設定した場合にのみ、「`watt.server.grpc.clientCertRequired` を「true」に設定してください。

注:
このパラメータの変更を適用するには、gRPC チャネルを再起動する必要があります。gRPC チャネルを再起動するには、無効にした後、有効にします。

watt.server.grpc.enable.TLS
Integration Server に組み込まれた gRPC サーバがデータ暗号化に TLS を使用し、受信 gRPC 要求に対してクライアント認証を必須にするかどうかを指定します。「true」に設定した場合、gRPC サーバは転送時に TLS を使用してデータを暗号化します。受信要求は認証に対してクライアント認証を必須にすることがあります。gRPC サーバが認証を必須にするかどうかは、`watt.server.grpc.clientCertRequired` の値によって決まります。「false」に設定した場合、gRPC サーバは plaintext モードで動作します。gRPC サーバは TLS を使用してデータを暗号化せず、受信要求に対してクライアント認証による認証を実行しません。plaintext モードでは、gRPC サーバは、受信要求で指定された認証情報を含むクライアント要求を拒否します。plaintext モードは開発環境および初期のテスト環境では役立つことがありますが、実稼働環境では使用しないでください。Software AG は、実稼働環境では暗号化とクライアント認証を確実に行うため、`watt.server.grpc.enable.TLS` を「true」に設定することをお勧めします。デフォルトは「true」です。

注:
このパラメータの変更を適用するには、gRPC チャネルを再起動する必要があります。gRPC チャネルを再起動するには、無効にした後、有効にします。

watt.server.grpc.maxConcurrentCalls
Integration Server に組み込まれた gRPC サーバが同時に処理可能な gRPC 要求の最大数を指定します。各 gRPC 要求はサーバスレッドプールからスレッドを使用します。デフォルトは 100 です。

注:
このパラメータの変更を適用するには、gRPC チャネルを再起動する必要があります。gRPC チャネルを再起動するには、無効にした後、有効にします。

watt.server.grpc.maxMessageSize
受信 gRPC 要求のメッセージの最大サイズ (バイト単位) を指定します。Integration Server に組み込まれた gRPC サーバは、この制限以下のサイズのメッセージを受け入れます。gRPC サーバはこの上限を超えるサイズのメッセージを拒否します。このパラメータは受信要求にのみ影響します。gRPC 応答には適用されません。デフォルトは 4,194,304 (4 MB) です。

注:
このパラメータの変更を適用するには、gRPC チャネルを再起動する必要があります。gRPC チャネルを再起動するには、無効にした後、有効にします。

watt.server.hostAccessMode
カスタム IP アクセスが設定されていないポートに対して IP アクセスを指定します。このパラメータが「`include`」に設定されている場合は、Integration Server Administrator インタフェースで明示的に拒否された IP アドレスを除くすべての IP アドレスからの要求を受け入

れます。このパラメータが「exclude」に設定されている場合は、Integration Server Administrator インタフェースで明示的に許可された IP アドレスを除くすべての IP アドレスからの要求を拒否します。デフォルトは「include」です。

watt.server.hostAllow
許可されたサービスであるホストの名前を指定します。デフォルトはありません。

watt.server.hotDeploymentAutoRecover
Integration Server で自動回復がパッケージのホット展開に対して有効かどうかを指定します。「true」に設定すると、パッケージの新しいバージョンのホット展開が失敗した場合に、パッケージの前バージョンの自動回復が有効になります。「false」に設定すると、ホット展開の失敗時のパッケージの自動回復が無効になります。デフォルトは「false」です。

watt.server.hotDepolymentAutoRecover の値は、Integration Server Administrator の [**設定**] > [ホット展開] > [設定の編集] ページの [**自動回復**] フィールドに関連付けられています。

watt.server.hotDeploymentTimeout
Integration Server がパッケージのホット展開を試行する前に実行中の処理の完了を待機する秒数を指定します。実行に時間のかかるタスクをパッケージの展開前に完了させる場合、タイムアウト値はタスクの完了にかかる時間よりも大きい値に設定します。デフォルトは 60 秒です。

watt.server.hotDeploymentTimeout の値は、Integration Server Administrator の [**設定**] > [ホット展開] > [設定の編集] ページの [**タイムアウト**] フィールドに関連付けられています。

watt.server.http.allowOptions
Integration Server でサポートされている HTTP メソッドを確認するために、Integration Server がクライアントに HTTP OPTIONS メソッドの使用を許可するかどうかを指定します。このパラメータを「true」に設定した場合、Integration Server は OPTIONS メソッドを許可します。このパラメータを「false」 (またはその他の値) に設定した場合、OPTIONS 要求を送信したクライアントは「405 Method Not Allowed」応答を受け取ります。

デフォルトは「true」です。

watt.server.http.authorizationEncoding
Integration Server のユーザ名およびパスワードをエンコーディングする場合に HTTP クライアント (ブラウザなど) が使用するエンコーディングを指定します。HTTP クライアントは、HTTP 要求ヘッダーの Authorization 属性でユーザ名およびパスワードを Integration Server に送信します。watt.server.http.authorizationEncoding プロパティを、ブラウザが Authorization 属性に使用するエンコーディングに設定します。ユーザ環境内で使用されるエンコーディングを確認するには、ブラウザのマニュアルを参照するか、ブラウザのベンダーに問い合わせてください。デフォルトは「UTF-8」です。

> **注:**
> 2 つのブラウザが Authorization 属性に異なるエンコーディングを使用しており、両方が、ASCII 以外のパスワードを Integration Server に送信した場合は、Authorization のエンコーディングが watt.server.http.authorizationEncoding プロパティの値に一致するブラウザのみが機能します。

watt.server.http.Content-Security-Policy
応答の HTTP セキュリティヘッダー Content-Security-Policy を Integration Server Administrator にアクセスする要求に設定します。このパラメータを使用して、クロスサイトスクリプティング (XSS)、データインジェクションなどの攻撃を検出し回避することができます。

デフォルトで、このプロパティは「none」に設定されており、Content-Security-Policy ヘッダーは応答に含まれていません。Integration Server Administrator へのアクセスをセキュリティで保護するには、次のようにこのプロパティの値を「script-src 'self' 'unsafe-inline'」に設定します。

```
watt.server.http.Content-Security-Policy=script-src 'self' 'unsafe-inline'
```

関連情報については、watt.server.http.X-Permitted-Cross-Domain-Policies、watt.server.http.Strict-Transport-Security、watt.server.http.X-Content-Type-Options、および watt.server.http.X-XSS-Protection を参照してください。

watt.server.http.forwardableHeaders

Integration Server が受信要求から収集して送信要求に含めるヘッダー。Integration Server は、watt.server.http.forwardHeaders が「true」に設定されているときのみ、ヘッダーを収集して転送します。watt.server.http.forwardableHeaders が空の場合、Integration Server は watt.server.http.forwardHeaders が「true」に設定されている場合でもヘッダーを収集せず、転送もしません。

watt.server.http.forwardableHeaders のデフォルトは次のとおりです。
x-request-id,x-b3-traceid,x-b3-spanid,x-b3-parentspanid,x-b3-sampled,x-b3-flags,x-ot-span-context

watt.server.http.forwardHeaders

Integration Server が Istio Service Mesh および OpenTracing により使用されるヘッダーを収集することで、分散トレースをサポートするかどうかを指定します。「true」に設定されている場合、Integration Server は、受信要求から watt.server.http.forwardableHeaders に指定されているヘッダーを収集し、受信要求の結果として発生する送信要求に含めます。「false」に設定されている場合、Integration Server はは受信要求からヘッダーを収集せず、転送もしません。デフォルト値は「true」です。

watt.server.http.header.sameSite

IS からの HTTP 応答が Set-Cookie HTTP 応答ヘッダーの SameSite 属性を含むかどうか、また SameSite 属性が含まれる場合はその値を含むどうかを指定します。このパラメータの有効な値は None、Strict、Lax です。これらの値とその動作は HTTP 仕様に準拠します。

注:

- HTTP 標準には、Set-Cookie HTTP 応答ヘッダーの SameSite 属性が None である場合に、Secure 属性を指定する必要があると記述されています。HTTP 応答ヘッダーに Secure 属性を含めるには、サーバ設定パラメータ watt.server.http.header.useSecure を「true」に設定します。
- Integration Server Administrator で watt.server.http.header.sameSite を None に設定しようとする場合、watt.server.http.header.useSecure パラメータが「false」に設定されていると、Integration Server Administrator に警告メッセージが表示され、値を変更できません。
- server.cnf 設定ファイルで watt.server.http.header.sameSite または watt.server.http.header.useSecure パラメータの値を直接編集し、指定した設定が有効でない場合、Integration Server はログに警告を記録し、無効な設定は無視されます。Integration Server のログの詳細については、235 ページの「サーバログの設定」を参照してください。watt.server.http.header.sameSite の値が無視されると、SameSite 属性は Set-Cookie 応答ヘッダーに追加されません。
- pub.flow:setResponseHeader または pub.flow:setResponseHeaders サービスを使用して Set-Cookie ヘッダーを設定し、SameSite 属性を含めた場合、Integration Server は watt.server.http.header.sameSite パラメータの値を無視します。

watt.server.http.header.useHttpOnly
Integration Server で HTTP クライアントへの応答に Set-Cookie ヘッダーの HttpOnly 属性を含めるかどうかを指定します。このプロパティを「true」 (デフォルト) に設定した場合、Integration Server は HTTP クライアントへの HTTP 応答の Set-Cookie ヘッダーに HttpOnly 属性を含めます。このプロパティを「true」に設定すると、保護された Cookie にクライアントサイドスクリプトがアクセスできなくなるため、Software AG ではこの設定をお勧めします。HttpOnly 属性をサポートしていない HTTP クライアントを使用している場合は、このプロパティを「false」に設定してください。

watt.server.http.header.useSecure
Integration Server で HTTP/S クライアントへの応答に Set-Cookie ヘッダーの secure 属性を含めるかどうかを指定します。watt.server.http.header.useSecure プロパティを「true」 (デフォルト) に設定した場合、Integration Server は HTTP/S クライアントに対する HTTPS 応答の Set-Cookie ヘッダーに secure 属性を含めます。このプロパティを「true」に設定すると、クライアントからサーバに送信される Cookie が必ず暗号化されるようになるため、Software AG ではこの設定をお勧めします。

重要:
watt.server.http.header.sameSite パラメータの値は、Integration Server がこのパラメータをどのように使用するかに影響を与えます。詳細については、watt.server.http.header.sameSite を参照してください。

watt.server.http.interceptor.enabled
受信 HTTP インターセプタを Integration Server に対して有効にするかどうかを指定します。「true」に設定した場合、Integration Server は watt.server.http.interceptor.impl プロパティで指定された受信 HTTP インターセプタ実装を使用します。このプロパティのデフォルト値は「false」です。

注:
初めてこのプロパティを「true」に設定した場合、このプロパティの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.http.interceptor.impl
受信 HTTP インターセプタを実装する java クラスの完全修飾名を指定します。指定したクラスは com.softwareag.is.interceptor.HttpInterceptorIFC インタフェースを実装し、インタフェースで両方のメソッド (`preProcess` および `postProcess`) の実装を提供する必要があります。サーバクラスパスでコンパイル済みクラスが利用できるようにする必要があります。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.http.interceptor.outbound.enabled
送信 HTTP インターセプタを Integration Server に対して有効にするかどうかを指定します。「true」に設定した場合、Integration Server は watt.server.http.interceptor.outbound.impl プロパティで指定された送信 HTTP インターセプタ実装を使用します。このプロパティのデフォルト値は「false」です。

重要:
初めてこのプロパティを「true」に設定した場合、このプロパティの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.http.interceptor.outbound.impl
送信 HTTP インターセプタを実装する java クラスの完全修飾名を指定します。指定したクラスは com.softwareag.is.interceptor.HttpOutboundInterceptorIFC インタフェースを実装し、インタフェースで両方のメソッド (`preProcess` および `postProcess`) の実装を提供する必要があります。サーバクラスパスでコンパイル済みクラスが利用できるようにする必要があります。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.http.interceptor.preprocess.sizeLimit
プリプロセス中に受信 HTTP インターセプタが読み取るバイトの最大数を指定します。HTTPRequest の入力がこの値を超える場合、受信 HTTP インターセプタはこの上限まで読み取り、残りの入力ストリームは無視します。このプロパティは、N [KB|MB|GB] に設定します。N は、有効な任意の整数です。整数と測定単位の間にスペースを入れないでください。単位を指定しない場合、Integration Server では N 値をバイトとして扱います。デフォルト値は -1 です。受信 HTTP インターセプタに渡される要求バイトのサイズに制限がないことを示します。受信 HTTP インターセプタは入力ストリーム全体を読み取ります。

watt.server.http.jsonFormat
HTTP クライアント要求の JSON コンテンツを Integration Server で処理する方法を指定します。このプロパティを「`parsed`」(デフォルト) に設定した場合、Integration Server は JSON コンテンツをパイプライン変数に自動的に解析します。次にクライアントは JSON を使用してパイプラインを構成し、pub.json:jsonStreamToDocument を呼び出すことなく、既存のサービスを実行できます。

watt.server.http.jsonFormat を「`stream`」に設定した場合、Integration Server は JSON コンテンツを含む jsonStream 変数を、InputStream としてパイプラインに配置します。次に pub.json:jsonStreamToDocument を使用して、jsonStream をパイプライン変数に解析できます。

watt.server.http.jsonFormat を「`bytes`」に設定した場合、Integration Server は JSON の受信ストリームをバイト配列に配置します。

jsonFormat クエリーパラメータを要求 URI に追加することで、個々の要求で watt.server.http.jsonFormat に指定された動作を上書きできます。たとえば、watt.server.http.jsonFormat を「`parsed`」に設定した場合、クライアントはこの設定を上書きして、以下のように URI を入力することで、要求に対して `stream` の値を指定できます。

```
/invoke/myfolder/myservice?jsonFormat=stream
```

これにより、JSON コンテンツを含む InputStream として、要求がパイプライン内の *jsonStream* 変数に配置されます。

watt.server.http.listRequestVars
Integration Server で HTTP 要求内の重複するクエリートークンを処理する方法を指定します。

設定値	Integration Server
`always`	重複するトークンと重複しないトークンをリストに変換します。次に例を示します。

設定値	Integration Server

	`http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z`
	
	結果は次のとおりです。
	`INVOKE folder:service {arg1=X, arg1List=[X,Y], arg2=Z, arg2List=[Z]}`

`asNeeded`	重複するトークンをリストに変換します。重複しないトークンは変換されません。次に例を示します。
	`http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z`
	
	結果は次のとおりです。
	`INVOKE folder:service {arg1=X, arg1List=[X,Y], arg2=Z}`
	
	これがデフォルト設定です。

`error`	URL クエリー内で重複するトークンが見つかった場合、ServiceException をスローします。重複しないトークンは変換されません。次に例を示します。 `http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z`

`never`	URL クエリー内で見つかった重複するトークンを無視します。次に例を示します。
	`http://host:5555/folder/service?arg1=X&arg1=Y&arg2=Z`
	
	結果は次のとおりです。
	`INVOKE folder:service {arg1=X, arg2=Z}`

watt.server.http.preserveUriReservedChars

Integration Server が、URI パスを評価する前に、要求内のパーセントエンコーディングされた URI パスをデコードする必要があるかどうかを指定します。このプロパティを「`true`」(デフォルト) に設定すると、Integration Server は URI パスを評価するまで、空白を含むパーセントエンコーディングされた予約文字のデコードを保留します。「`false`」に設定すると、Integration Server は URI パスを評価する前に、空白を含むパーセントエンコーディングされた予約文字をデコードします。

注:
watt.server.http.preserveUriReservedChars の値は、rest ディレクティブを使用する REST 要求に対してのみ重要です。rest ディレクティブを使用する要求では、パスの最後の数個のトークンが、リソースのインスタンスを識別したり、アプリケーション固有の情報を提供したりします。これらのトークンは、パイプライン内で $resourceID 変数および $path 変数となります。この部分の URI パスには、空白を含むパーセントエンコーディングされた予約文字が含まれる場合があります。watt.server.http.preserveUriReservedChars を「true」に設定した場合、Integration Server は、URI パスを評価してからサービスを起動する前に、パーセントエンコーディングされた文字をデコードします。

invoke ディレクティブを使用する要求では、「/invoke」の後の URI の絶対パスは Integration Server ネームスペース内のフォルダとサービスを参照します。フォルダとサービスには空白を含む URI の予約文字を含めることはできません。そのため、invoke ディレクティブを使用し、空白を含むパーセントエンコーディングされた予約文字がパスに含まれている要求はすべて、実際の Integration Server サービスを識別できず、常に HTTP 404 応答を戻します。

watt.server.http.reauth.user-agent.list
HTTP セッションが期限切れになった後に、Integration Server で再認証を要求する HTTP クライアントをセミコロン区切りのリストで指定します。デフォルトでは、このパラメータは「`Firefox;MSIE`」に設定されています。そのため、Mozilla Firefox および Microsoft Internet Explorer の各ブラウザの HTTP セッションが期限切れになった場合、これらのブラウザからの要求の再認証が求められます。このリストに指定されていないユーザエージェントに対しては、Integration Server はセッションが期限切れになったときに新しいセッション Cookie を送信します。

このプロパティを削除し、Integration Server を再起動すると、このプロパティの値は「`Firefox;MSIE`」にリセットされます。

watt.server.http.request.supportCompression
Integration Server が HTTP 要求の圧縮をサポートする必要があるかどうかを指定します。デフォルトでは、このパラメータは「`false`」に設定されます。Integration Server が HTTP 要求で圧縮されたペイロードを受信した後にペイロードを圧縮解除することをサポートする場合、このパラメータを「`true`」に設定します。HTTP 要求の圧縮の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

Integration Server は、HTTP 要求で定義された `Content-Encoding` 要求ヘッダーに基づいてペイロードを圧縮解除します。要求ヘッダーは、ペイロードの圧縮に使用された圧縮スキームを定義します。Integration Server は同じ圧縮スキームを使用してペイロードを圧縮解除します。サポートされている圧縮スキームは ***Gzip*** および ***Deflate*** です。

パラメータが「`false`」に設定された状態で Integration Server が HTTP 要求で圧縮されたペイロードを受信した場合、その要求は無視されます。

watt.server.http.response.supportCompression
Integration Server が HTTP 応答の圧縮をサポートする必要があるかどうかを指定します。デフォルトでは、このパラメータは「`false`」に設定されます。Integration Server が HTTP 応答を送信する前にペイロードを圧縮することをサポートする場合、このパラメータを「`true`」に設定します。HTTP 応答の圧縮の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

Integration Server は、HTTP 要求で定義された `Accept-Encoding` 要求ヘッダーに基づいてペイロードを圧縮します。要求ヘッダーは、Integration Server がペイロードの圧縮に使用できる推奨圧縮スキームを定義します。サポートされている圧縮スキームは ***Gzip*** および ***Deflate*** です。

パラメータが「`false`」に設定された状態で Integration Server が応答のペイロードを圧縮する HTTP 要求を受信した場合、圧縮されていない応答を送信します。

watt.server.http.returnException
呼び出されたサービスがエラーで終了したときに、Integration Server で HTTP/SOAP 応答の送信時にスタックトレースを呼び出し元クライアントに返すかどうかを指定します。次のいずれかの値を指定できます。

値	Integration Server の動作
`true`	HTTP/SOAP 応答でスタックトレースを返します。これがデフォルトです。
`false`	スタックトレースを返しません。
`webMethods`	クライアントの HTTP ユーザエージェントが webMethods に設定されているか、または watt.net.userAgent パラメータで指定されたユーザエージェントと値が同じ場合にのみ、スタックトレースを返します。

watt.server.http.securityRealm
Integration Server セキュリティ領域の名前を指定します。この値は、認証を要求するクライアントに対して Integration Server が送信する HTTP 応答のヘッダーに含められます。デフォルト値は「**Integration Server**」です。

watt.server.http.Strict-Transport-Security
Integration Server が応答ヘッダーに Strict-Transport-Security ヘッダーを含めるかどうかを指定します。このパラメータに指定した値は Integration Server Administrator にアクセスする Web 要求の応答ヘッダーに設定されます。ユーザはこのパラメータに任意の有効な値を設定できます。値を「*None*」に設定するか空白のままにすると、応答ヘッダーに Strict-Transport-Security ヘッダーは含まれません。

例: watt.server.http.Strict-Transport-Security=*max-age=300; includeSubDomains*

このパラメータの有効な値の詳細については、『*Internet Engineering Task Force (IETF) November 2012 specification*』の「HTTP Strict Transport Security (HSTS)」セクションを参照してください。

注:
Web サーバは HTTP Strict-Transport-Security 応答ヘッダー (HSTS) を使用して、Web ブラウザが HTTP 接続ではなくセキュアな HTTPS 接続を使用してのみ Web サーバにアクセスする必要があることを通知できます。

watt.server.http.uriPath.decodePlus
「+」文字が要求 URI のパスコンポーネントに使用された場合、「 」文字として解釈されるかどうかを指定します。「true」に設定すると (デフォルト)、Integration Server では要求 URI を処理するときに「+」を「 」に変換します。「false」に設定すると、Integration Server では要求 URI を処理するときに「+」を「 」に変換しません。このパラメータは、要求 URI のパスコンポーネントにある「+」文字の処理にのみ影響を与えます。
watt.server.htt.uriPath.decodePlus の値と関係なく、要求 URI のクエリーに使用される「+」文字はすべて「 」文字に変換されます。

watt.server.http.useAcceptHeader
Integration Server で HTTP 要求の承認ヘッダーフィールドをサポートするかどうかを指定します。承認ヘッダーフィールドには、HTTP クライアントが HTTP 応答で受け入れることになる 1 つまたは複数のコンテンツタイプが指定されています。デフォルト設定は「`true`」です。watt.server.content.type.mappings パラメータを使用して、承認ヘッダーフィールド内のワイルドカードのコンテンツタイプを特定のコンテンツタイプにマッピングする場合は、このパラメータを「`true`」に設定する必要があります。

watt.server.http.X-Content-Type-Options

Integration Server が応答ヘッダーに X-Content-Type-Options ヘッダーを含めるかどうかを指定します。このパラメータに指定した値は Integration Server Administrator にアクセスする Web 要求の応答ヘッダーに設定されます。デフォルト値は「None」です。このパラメータには任意の有効な値を指定できます。値を「*None*」に設定するか空白のままにすると、応答ヘッダーに X-Content-Type-Options ヘッダーは含まれません。

例: watt.server.http.X-Content-Type-Options=*nosniff*

このパラメータの有効な値の詳細については、*https://fetch.spec.whatwg.org/#x-content-type-options-header* の「X-Content-Type-Options」セクションを参照してください。

注:
Integration Server の応答ヘッダーに含まれる X-Content-Type-Options ヘッダーは、Content-Type ヘッダーで通知された MIME タイプに続くことを示します。

watt.server.http.x-frame-options

Integration Server が応答ヘッダーの X-Frame-Options 属性を処理する方法を制御します。デフォルトでは、パラメータは「**SAMEORIGIN**」に設定されていて、クライアントのブラウザは、フレームが同じサーバのページにある場合のみ、Integration Server ページが HTML フレーム内に表示されることを許可します。

注:
Integration Server は、HTTP ヘッダー Field X-Frame-Options の仕様で定義されているように、Web ページの要求に対する応答ヘッダーに X-Frame-Options 属性を含みます。X-Frame-Options は、invoke、rest、restv2、または soap ディレクティブといったサービス呼び出しの要求に対する応答には含まれません。
https://my-server/MyPackage/my-page.html といった Web ページの要求に対する応答にのみ含まれます。

Integration Server が応答ヘッダーに X-Frame-Options 属性を含めないようにするには、**watt.server.http.x-frame-options** プロパティから値を削除します。たとえば、watt.server.http.x-frame-options= にします。このプロパティは、Integration Server Administrator の **[設定] > [拡張設定]** ページで設定できます。このプロパティに対する変更はただちに有効になり、サーバを再起動する必要はありません。

注:
値 DENY は X-Frame-Options 属性に定義されていますが、Integration Server に対しては許可されません。DENY は、フレームの作成元に関係なく、ページをフレーム内に表示できないということを意味します。このため、Integration Server Administrator は使用できなくなります。**watt.server.http.x-frame-options** が DENY に設定された場合、この値は無視され、代わりに SAMEORIGIN が使用されます。

注:
「`ALLOW-FROM other_origin`」の値は、許可されていますが、ブラウザに無視されます。Software AG は watt.server.http.x-frame-options を「`ALLOW-FROM other_origin.`」に設定することを推奨していません。「`ALLOW-FROM other_origin`」を使用すると、クライアントブラウザは、フレームされたコンテンツの元のサーバとは関係なく、Integration Server ページが HTML フレーム内に表示されることを許可します。Integration Server は、X-frame-options ヘッダーを応答に含めません。

watt.server.http.X-Permitted-Cross-Domain-Policies
HTTP セキュリティヘッダー X-Permitted-Cross-Domain-Policies を設定します。このヘッダーは、Integration Server Administrator へのアクセスに使用できるドメイン間ポリシーをクライアントに通知します。デフォルトで、このプロパティは空であり、X-Permitted-Cross-Domain-Policies ヘッダーは応答に含まれていません。Integration Server Administrator へのアクセスをセキュリティで保護するため、このプロパティを「none」に設定できます。この場合、クライアントによるドメインからのデータのロードが停止されます。

関連情報については、watt.server.http.Content-Security-Policy、watt.server.http.Strict-Transport-Security、watt.server.http.X-Content-Type-Options、および watt.server.http.X-XSS-Protection を参照してください。

watt.server.http.X-XSS-Protection
Integration Server が応答ヘッダーに X-XSS-Protection を含めるかどうかを指定します。このパラメータに指定した値は Integration Server Administrator にアクセスする Web 要求の応答ヘッダーに設定されます。デフォルト値は「None」です。このパラメータには任意の有効な値を指定できます。値を「*None*」に設定するか空白のままにすると、応答ヘッダーに X-XSS-Protection は含まれません。

例: watt.server.http.X-XSS-Protection=*1*; *mode=block*

このパラメータの有効な値の詳細については、ブラウザのマニュアルを参照してください。

> **注:**
>
> HTTP X-XSS-Protection 応答ヘッダーは、反射型クロスサイトスクリプティング (XSS) 攻撃を検出すると、Web ページのロードを停止します。

watt.server.http.xmlFormat
Integration Server で受信 XML ドキュメントを自動的に解析するか、それらをサービスに直接渡すかを指定します。Integration Server で HTTP を介して XML ドキュメントを渡す方法についてデフォルトの動作を設定するには、このプロパティを使用します。次のいずれかの値を指定します。

指定内容	動作
`bytes`	Integration Server は、解析を省略して、要求されたサービスに XML バイトを直接渡します。このパラメータを「bytes」に設定した場合、XML はサービスの入力パイプラインに「*xmlBytes*」という名前のバイト配列として表示されます。
`enhanced`	Integration Server では、拡張 XML パーサを使用して XML を自動的に解析し、org.w3c.dom.Node インタフェースを実装した *node* という名前の拡張ノードオブジェクトとして、その XML をサービスに渡します。
`node`	Integration Server は、レガシー XML パーサを使用して XML を自動的に解析し、*node* という名前の com.wm.lang.xml.Node タイプのパラメータでサービスにその XML を渡します。これがデフォルトの動作です。
`stream`	Integration Server は、解析を省略して、要求されたサービスに XML ストリームを直接渡します。このパラメータを「stream」に設定した場合、XML

指定内容	動作
	はサービスの入力パイプラインに「*xmlStream*」という名前の InputStream として表示されます。

watt.server.http.xmlFormat の値を指定しないか、または値が無効な場合、XML 解析のデフォルトの動作は「node」になります。

この機能は、HTTP または HTTPS のみを介して XML をサブミットまたは受信するときに使用できます。

注:
このパラメータで設定されたデフォルトは、個々のクライアント要求で URL に xmlFormat 引数を使用すると上書きされます。xmlFormat 引数の詳細については、『*webMethods Service Development Help*』を参照してください。

注:
すべての Integration Server に対するデフォルトの xmlFormat を変更すると、既存のサービスおよびアプリケーションが失敗する可能性があります。サービスに対する [**デフォルトの xmlFormat**] プロパティを使用するのではなく、サービスまたはアプリケーションベースの xmlFormat の変更を検討してください。[**デフォルトの xmlFormat**] プロパティの詳細については、『*webMethods Service Development Help*』を参照してください。

watt.server.httplog
Integration Server が HTTP ログを書き込むかどうか、およびログファイルにどの形式を使用するかを指定します。パラメータは、次のいずれかの値に設定します。

値	説明
`common`	Integration Server は共通ログ形式に準拠した HTTP ログエントリを書き込みます。
`combined`	Integration Server は複合ログ形式に準拠した HTTP ログエントリを書き込みます。
`custom`	Integration Server はレガシー形式で HTTP ログエントリを書き込みます。

Integration Server 10.11 より前のバージョンでは、`watt.server.httplog` に指定できる値は「`true`」または「`false`」でした。これらの値は今でも使用できます。「true」に設定した場合、Integration Server は、この修正前に HTTP ログ処理に使用されていたレガシー形式で HTTP ログエントリを書き込みます。「false」に設定した場合、Integration Server は HTTP ログ処理を実行しません。

`watt.server.httplog` が common、combined、custom、または true 以外の値に設定されているか、値が設定されていない場合、Integration Server は HTTP ログ処理を実行しません。

`watt.server.httplog` のデフォルト値はありません。つまり、デフォルトで、HTTP ログ処理は実行されません。

watt.server.hostDeny
拒否されたサービスであるホストの名前を指定します。デフォルトはありません。

watt.server.idatacoder.legacy
Integration Server は、バイナリプロトコルを使用したドキュメントのエンコードに IDataBinCoder の現在のバージョンを使用するかレガシーバージョンを使用するかを指定します。レガシーバージョンを使用するには、「true」に設定します。新しいバージョンを使用するには、「false」に設定します。大きく複雑なドキュメントに対し、こちらの方がはるかに高速になり、メモリの使用が多少効率的になることがあります。デフォルトは「false」です。

注:
`watt.server.idatacoder.legacy` の値が、ドキュメントのエンコード時からドキュメントのデコードが必要なときまでの間に変更された場合でも、ディスクにパーシストされたドキュメントはデコードできます。つまり、エンコードに使用された IDataBinCoder のバージョンに関係なく、パーシストされたデータはデコードできます。

watt.server.idr.reaperInterval
スケジュール済みのシステムサービスのメッセージ履歴スイーパを実行し、期限切れドキュメント履歴の削除を実行する間隔を分単位で指定します。デフォルトは 10 分です。

watt.server.illegalNSChars
パッケージ、フォルダまたはサービスに名前を付けるときに使用できない文字を指定します。デフォルトは `?`-#&@^!%*:$.\/';,~+=)(|}{][><` です。

watt.server.illegalUserChars
ユーザ名で使用できない文字を指定します。デフォルトは「`"\<&`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

重要:
このプロパティから文字を削除しないでください。文字を削除して、その文字を含むユーザを作成した場合は、Integration Server が users.cnf を解析できず、起動しなくなります。

watt.server.inetaddress
サーバが受信要求を受信待機する IP アドレスを指定します。デフォルトでは、Integration Server は使用可能なすべての IP アドレスで受信待機します。サーバが単一の IP アドレスで受信待機するように制限するには、このプロパティの値としてその IP アドレスを指定します。

watt.server.invokeDirective
Integration Server のサービスを呼び出す URL の invoke ディレクティブに使用する代替語を指定します。デフォルトでは、このパラメータは `watt.server.invokeDirective=invoke` と設定され、ユーザは invoke ディレクティブを invoke (`http://`*`host`*`:`*`port`*`/invoke/`*`folder`*`/`*`service_name`*) と指定する必要があります。invoke ディレクティブを `invoke` または代替語で指定することをユーザに許可する場合は、このパラメータに代替語を設定します。たとえば、invoke ディレクティブを `invoke` または `submit` のいずれかで指定することをユーザに許可する場合は (`http://`*`host`*`:`*`port`*`/invoke/`*`folder`*`/`*`service_name` または `http://`*`host`*`:`*`port`*`/submit/`*`folder`*`/`*`service_name`)、このパラメータを `watt.server.invokeDirective=submit` と設定します。

watt.server.invoke.maxRetryPeriod
Integration Server が行ったサービスの再試行が最大回数に達した場合に経過する待機時間の合計 (ミリ秒単位) を指定します。デフォルトは 15,000 ミリ秒 (15 秒) です。個々のサービスについて再試行回数を設定するときは、[**最大試行回数**] の値を [**再試行間隔**] で乗算して計算した値が、このサーバパラメータで設定されている値を超えてはなりません。サービスの再試行を設定する方法の詳細については、『*webMethods Service Development Help*』を参照してください。

watt.server.java.unicode
Java サービスのソースコードが Unicode エンコーディングで保存されているかどうかを指定します。デフォルトは「`false`」です。サーバのネイティブエンコーディングに変換できない文字がソースコードに含まれる場合は、「`true`」に設定します。

watt.server.jca.connectionPool.thresholdWaitingRequest
このプロパティを有効化すると、接続プールの最大接続数として設定された値 ([接続] ページの [最大プールサイズ] パラメータを使用して設定した値) を上増しするパーセンテージを表すことができます。たとえば、このプロパティを `watt.server.jca.connectionPool.thresholdWaitingRequest=20` と設定すると、設定済みの最大接続数の 120% というしきい値が設定されます。

このプロパティを定義しないか、ゼロよりも小さい値を指定した場合、この機能は無効化されます。

このプロパティを有効化した場合、接続プール内の待機中の接続要求と使用中の接続数は、しきい値を超えないレベルに維持されます。

watt.server.jca.transaction.rollbackOnWriteFailure
Integration Server がトランザクションの状態とその参加リソースを (ストアが損傷しているなどの理由で) XA 回復ストアに格納できない場合に、Integration Server がトランザクションを続行するか (`false`)、ロールバックするか (`true`) を指定します。このパラメータを「`false`」に設定するには、多少のリスクが伴います。この場合、Integration Server が異常終了しても XA 回復ストアに状態が保存されないため、Integration Server は未完了トランザクションを解決できないほか、未完了トランザクションを手動で解決することもできません。

watt.server.jca.transaction.writeRecoveryRecord
Integration Server が、XA トランザクション回復で使用するために XA トランザクションの情報を保持するかどうかを指定します。Integration Server が XA トランザクションの情報を保存しない場合は、Integration Server を使用して未完了の XA トランザクションを回復することはできません。つまり、Integration Server が未完了の XA トランザクションを自動的に回復する処理を行わないため、Integration Server Administrator を使用して未完了トランザクションを手動で回復または解決することができません。XA トランザクション回復を有効にするには、「`true`」と指定します。XA トランザクション回復を無効にするには、「`false`」と指定します。デフォルトは「`true`」です。

watt.server.jdbc.datadirect.snoop.default
DataDirect Connect JDBC ドライバ向けに、DataDirect Snoop ツールのデフォルト設定を指定します。DataDirect Snoop ツールは、Integration Server と外部 RDBMS の間のネットワークパケットをログに記録します。作成されたログファイルは、トレースおよび診断の目的で使用することができます。

注:

このオプションは、外部 RDBMS のみに使用します。組み込みの IS 内部データベースには使用できません。

データのログを記録する DataDirect Connect JDBC ドライバごとにこのパラメータを設定します。

重要:
この機能を有効にして Integration Server で使用するには、[**Snoop**] オプションを Integration Server Administrator の [**JDBC 接続プールエイリアスの編集**] ページまたは [**JDBC 接続プールエイリアスの作成**] ページで選択する必要があります。

このパラメータのデフォルト値は以下のとおりです。

```
ddtdbg.ProtocolTraceEnable=true;ddtdbg.ProtocolTraceMaxline=16;
      ddtdbg.ProtocolTraceLocation=/logs/snoop/<alias_name>.log;
      ddtdbg.ProtocolTraceShowTime=true
```

alias_name は JDBC 接続プールエイリアスの名前です。

注:
DB2 の場合は、値の末尾に以下の値を含めます。

```
ddtdbg.ProtocolTraceEBCDIC=true
```

デフォルト値では、ログファイルの名前と場所、および DataDirect Snoop ツールの属性を定義します。通常、デフォルト値を変更する必要はありません。ただし、属性がシステムのニーズを満たさない場合は、値を変更することができます。ログファイルの場所を変更する場合は、診断ツールが *Integration Server_directory* /instances/*instance_name*/logs/snoop ディレクトリからデータを収集することに注意してください。ログファイルの場所を変更すると、DataDirect Snoop ツールによって収集されるデータを診断ユーティリティがインポートできないことがあります。診断ユーティリティの使用方法の詳細については、995 ページの「診断ユーティリティの使用方法」を参照してください。DataDirect Snoop ツールの属性を設定する方法の詳細については、DataDirect Web サイトでマニュアルを参照してください。

重要:
このパラメータの値を変更する場合、接続プールを再起動して変更内容を有効にする必要があります。

watt.server.jdbc.datadirect.spy.default
DataDirect Connect JDBC ドライバについて、DataDirect Spy 診断機能のデフォルト設定を指定します。DataDirect Spy は、Integration Server と外部 RDBMS との間の JDBC 呼び出しおよび SQL ステートメントをログに記録します。作成されたログファイルは、トレースおよび診断の目的で使用することができます。

注:
このオプションは、外部 RDBMS のみに使用します。組み込みの IS 内部データベースには使用できません。

データのログを記録する DataDirect Connect JDBC ドライバごとにこのパラメータを設定します。

重要:

この機能を有効にして Integration Server で使用するには、[**Spy**] オプションを Integration Server Administrator の [**JDBC 接続プールエイリアスの編集**] ページまたは [**JDBC 接続プールエイリアスの作成**] ページで選択する必要があります。

このパラメータのデフォルト値は以下のとおりです。

```
SpyAttributes=(log=(file)/logs/spy/<alias_name>.log;logTName=yes;timestamp=yes)
```

alias_name は JDBC 接続プールエイリアスの名前です。

デフォルト値では、ログファイルの名前と場所、および DataDirect Spy の属性を定義します。通常、デフォルト値を変更する必要はありません。ただし、属性がシステムのニーズを満たさない場合は、値を変更することができます。ログファイルの場所を変更する場合は、診断ツールが *Integration Server_directory* /instances/*instance_name*/logs/spy ディレクトリからデータを収集することに注意してください。ログファイルの場所を変更すると、DataDirect Spy によって収集されるデータを診断ユーティリティがインポートできないことがあります。診断ユーティリティの詳細については、995 ページの「診断ユーティリティの使用方法」を参照してください。DataDirect Spy の属性を設定する方法の詳細については、DataDirect Web サイトでマニュアルを参照してください。

重要:
このパラメータの設定を変更する場合、接続プールを再起動して変更内容を有効にする必要があります。

watt.server.jdbc.datadirect.useJaasSubjectForKerberos
Integration Server が JAAS 設定の Subject を使用してデータベースに接続を確立するかどうかを指定します。「true」に設定した場合、JDBC プール接続でケルベロスを使用するときに Integration Server は JAAS 設定の Subject を使用します。Integration Server の JAAS 設定が別のコンポーネントによって消去される JAAS 競合がある場合、「true」に設定します。「false」に設定した場合、Integration Server は従来の方法でデータベース接続を確立しようとします。JAAS 競合がない場合、「false」に設定します。デフォルト値は「false」です。JAAS 競合の発生により、“No LoginModules configured for *loginConfigName from database URL*” というメッセージが表示され、JAAS ログインモジュールが JDBC 接続プールとして正しく設定されている場合、パラメータを「true」に設定します。

watt.server.jdbc.defaultDriver
WmDB で使用します。ドライバ名がデータベースエイリアスに指定されていない場合に、データベースへの接続に使用するドライバの Java クラスの名前を指定します。

watt.server.jdbc.derby.commitTolerance
Integration Server で関連する接続を接続プールから削除する前に、組み込みデータベースへのコミットが完了するのを待機する時間をミリ秒単位で指定します。ISInternal、DocumentHistory および Xref 機能のいずれかに組み込みデータベースを使用するように Integration Server が設定されている場合、このプロパティが使用されます。このプロパティで定義されている値よりコミットプロセスに時間がかかる場合、Integration Server は関連する接続を接続プールから削除し、今後の要求には新しい接続を開始できるようにします。このプロパティのデフォルト値は 150 ミリ秒です。

watt.server.jdbc.driverList
サーバの初期化時にロードする JDBC ドライバをカンマ区切りのリストで指定します。デフォルトはありません。

注:

このパラメータは、WmDB パッケージにのみ使用します。

watt.server.jdbcLogFile
JDBC ログアクティビティを格納するログファイルの名前を指定します。watt.server.jdbcLogging プロパティを有効にすると (「on」に設定すると)、このファイルにアクティビティが記録されます。デフォルト値は「jdbc.log」です。このファイルは、 *Integration Server_directory* \instances*instance_name*\logs フォルダに配置されます。ログファイルが生成されるのは、WmDB がデータベースへの接続を確立した後です。

注:
このパラメータは、WmDB パッケージにのみ使用します。

watt.server.jdbc.loginTimeout
Integration Server 上の JDBC ドライバがデータベースへの接続試行中に応答を待機する最大時間を秒単位で設定します。Integration Server で割り当て時間内に応答を受信しなければ、要求は終了され、エラーが記録され、別の処理に切り替わります。デフォルト値は 60 秒です。

watt.server.jdbc.loginTimeout で指定された秒数以内に Integration Server が監査ログデータベースに接続できない場合、Integration Server はすべての接続を ISCoreAudit JDBC 接続プールから削除して、接続を再作成します。接続の問題が迅速に解決された場合 (一時的なネットワークの障害など)、Integration Server は監査ログデータベースに再接続します。Integration Server が監査ログデータベースに再接続できない場合 (データベースがオフラインである場合など)、Integration Server はサーバログに例外を書き込みます。

重要:
このパラメータの設定を変更する場合、ISCoreAudit JDBC 接続プールを再起動して変更内容を有効にする必要があります。プールを再起動するには、Integration Server Administrator で [**設定**] > [JDBC プール] ページに移動して、ISCoreAudit 機能エイリアスの [**再起動**] をクリックします。

watt.server.jdbcLogging
Integration Server で java.sql.DriverManager に対するログを有効にして、JDBC アクティビティが記録されるようにするかどうかを指定します。「on」に設定した場合、watt.server.jdbcLogFile で指定したファイルにログファイルデータが書き込まれます。デフォルトは「off」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

注:
このパラメータは、WmDB パッケージにのみ使用します。

watt.server.jdbc.moreResults
Integration Server で 1 つ結果セットからのストアドプロシージャの結果を処理するか、複数の結果セットからのストアドプロシージャの結果を処理するかを指定します。「true」に設定した場合、Integration Server は複数の結果セットから生成された情報を処理して返します。「false」に設定した場合、Integration Server は 1 つの結果セットから生成された情報を処理します。デフォルトは「false」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

注:
このパラメータは、WmDB パッケージにのみ使用します。

watt.server.jdbc.sp.mandateParams
pub.db:call サービスからストアドプロシージャを呼び出すときに、pub.db:call サービスによって *$data* 入力パラメータの値が求められるようにするかどうかを指定します。*$data* 入力パラメータの値を指定しない場合、このプロパティを「`false`」に設定します。pub.db:call サービスによって *$data* 入力パラメータの値が求められるようにする場合、このプロパティを「`true`」に設定します。デフォルトは「`true`」です。

watt.server.jdbc.statementCache
Integration Server で作成済みのステートメントをキャッシュして後で使用できるようにするかどうかを指定します。「true」に設定した場合、サーバは作成済みのステートメントをローカルキャッシュに保存します。サーバは、データベースに対する後続の要求を受信したときに、キャッシュ内の作成済みのステートメントを再利用できるため、要求があるたびにステートメントを再作成する必要はありません。Java ヒープ領域を効率的に使用するために、30 秒以内にキャッシュ内のステートメントが再利用されなければ、そのステートメントは自動的に削除されます。デフォルトは「`false`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

注:
このパラメータは、WmDB パッケージにのみ使用します。

watt.server.jms.csq.maxRedeliveryCount
クライアントサイドキューからドキュメントの再デリバーを試行する回数を指定します。正の整数値を指定します。デフォルトは 5 です。設定された試行回数で Integration Server がドキュメントをデリバーできない場合、そのドキュメントはクライアントサイドキューから削除され、デリバーされません。この場合は、Integration Server によって JMSDeliveryFailureEvent がスローされます。

注:
このプロパティに大きい値を指定すると、Broker が使用不可であるなど、ドキュメントのデリバー失敗が一時的なエラーではない場合に Integration Server のパフォーマンスに影響する可能性があります。ドキュメント自体の問題によってドキュメントをデリバーできない場合は、設定されている回数までドキュメントの再デリバーが繰り返されるため、Integration Server のパフォーマンスに影響を及ぼします。

注:
このパラメータは、JMS プロバイダに関係なく、JMS メッセージングに適用されます。Broker がメッセージングプロバイダである webMethods メッセージングで使用するプロパティは watt.server.publish.maxCSQRedeliveryCount です。Universal Messaging がメッ

セージングプロバイダである webMethods メッセージングで使用するプロパティは watt.server.messaging.csq.maxRedeliveryCount です。

watt.server.jms.csq.publishDelayWhileDraining
Integration Server がクライアントサイドキュー (CSQ) に JMS メッセージをパブリッシュする前に CSQ の排出中に待機する秒数を指定します。CSQ に 1 つ以上のメッセージが含まれているときに遅延を使用するには、0 より大きく 60 以下の整数を指定します。

CSQ 内のメッセージ数によって遅延の長さを変化させるよう Integration Server に指示するには、0 を指定します。watt.server.jms.csq.publishDelayWhileDraining を 0 に設定した場合、Integration Server の待機時間は次のとおりです。

- CSQ のメッセージ数が 1～1000 の場合、6 秒
- CSQ のメッセージ数が 1001～5000 の場合、8 秒
- CSQ のメッセージ数が 5000 を超える場合、10 秒

watt.server.jms.csq.publishDelayWhileDraining は 0 です。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.connection.monitorPeriod
Integration Server で JMS プロバイダへの接続の状態をチェックする頻度 (秒単位) で指定します。最小値は 1 秒です。このパラメータを 1 未満に設定した場合、Integration Server は代わりにデフォルト値を使用します。デフォルトは 45 秒です。

watt.server.jms.connection.pingDestination
JMS プロバイダ上の送信先にキープアライブメッセージを送信するように Integration Server を設定して、Integration Server とサードパーティの JMS プロバイダ間の接続がファイアウォールによって終了されないようにします。キープアライブメッセージとは、即時のタイムアウトが設定された空白の非永続 JMS メッセージです。

このパラメータは、次のいずれかの値に設定します。

指定内容	[宛先]
`blank`	Integration Server と JMS プロバイダ間の接続を使用して JMS セッションの作成を試みます。これがデフォルトです。
`temp`	JMS プロバイダ上に一時キューを作成し、その一時キューにキープアライブメッセージを送信します。temp の大文字と小文字は区別されません。
`<destinationName>`	JMS プロバイダ上の指定した送信先にキープアライブメッセージを送信します。指定した送信先が JMS プロバイダ上に存在しない場合、Integration Server は Integration Server と JMS プロバイダ間の接続を使用して定期的に JMS セッションの作成を試みます。

注:
Integration Server でキープアライブメッセージを送信する頻度は、watt.server.jms.connection.pingPeriod の値によって決まります。

watt.server.jms.connection.pingPeriod
Integration Server で JMS プロバイダに ping を実行する頻度を秒単位で指定します。ping は、JMS プロバイダに送信されるキープアライブ要求として機能します。デフォルトは 300 秒 (5 分) です。

watt.server.jms.connection.retryPeriod
Integration Server で JMS プロバイダまたは JNDI プロバイダへの接続が失敗した場合の接続試行間の待機時間を秒単位で指定します。デフォルトは 20 秒です。

watt.server.jms.connection.update.blockingTime
pub.jms* サービスで使用される接続の更新中に、対象のサービスがその接続を待機する最大時間をミリ秒単位で指定します。ブロック時間が経過する前に Integration Server が接続を再開できない場合、Integration Server は ISRuntimeException をスローし、送信サービスは失敗します。値として 0 (ゼロ) を指定すると、Integration Server は JMS 接続エイリアスの更新中に pub.jms* サービスをブロックしません。この場合、Integration Server は即時に ISRuntimeException をスローします。最大値は 10,000 ミリ秒 (10 秒) です。デフォルトは 1000 ミリ秒です。

無効な値を入力した場合、Integration Server は watt.server.jms.connection.update.blockingTime パラメータをデフォルト値にリセットします。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.connection.update.restartDelay
JMS メッセージを送信するサービスで使用される接続を更新する必要がある場合に、Integration Server でそれらのサービスが処理を完了するのを待機する時間をミリ秒単位で指定します。再開遅延時間が経過したら、Integration Server は更新後の接続ファクトリオブジェクトを使用して接続をリフレッシュしてから、その接続を再開します。pub.jms* サービスが処理を完了する前に再開遅延時間が経過した場合、Integration Server は ISRuntimeException をスローし、送信サービスは失敗します。最大値は 10,000 ミリ秒 (10 秒) です。デフォルトは 500 ミリ秒です。

無効な値を入力した場合、Integration Server は watt.server.jms.connection.update.restartDelay パラメータをデフォルト値にリセットします。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.csq.batchProcessingSize
Integration Server でクライアントサイドキューから抽出して JMS プロバイダに送信する最大メッセージ数を指定します。JMS プロバイダへの接続が失敗すると、Integration Server は送信メッセージをクライアントサイドキューに入れます。JMS プロバイダに接続できるようになると、Integration Server はクライアントサイドキューの排出を開始します。デフォルトは 25 です。

watt.server.jms.guaranteedMultisend.alwaysUseTxLogging
Integration Server でマルチ送信保証付きポリシーに従って JMS メッセージを送信するときに、常に XA トランザクションロギングを使用するかどうかを指定します。XA トランザクションロギングでは、Integration Server は各トランザクションの状態を記録します。XA トランザクションロギングを使用すると、Integration Server の障害が原因で完了しなかったトランザクションを Integration Server で回復できるようになります。これはトランザクション

の整合性を保つ最も信頼できる方法ですが、パフォーマンスの点を考えると割高になることがあり、必ずしも必要ではありません。このプロパティを「true」に設定した場合、Integration Server は常に XA トランザクションロギングを使用し、接続トランザクションのタイプに関係なく、マルチキャスト保証付きポリシーに準拠した XA トランザクション回復を実行できます。「false」に設定した場合、Integration Server は接続トランザクションのタイプが XA_TRANSACTION である場合にのみ XA トランザクションロギングを使用します。デフォルトは「false」です。

watt.server.jms.nirvana.durableSubscriber.includeClientId
JMS 接続エイリアスのクライアント ID を、Designer を使用して作成された継続的サブスクライバの名前のプリフィックスとして使用するかどうかを指定します。JMS 接続エイリアスの [**クライアント ID**] プロパティの値を継続的サブスクライバ名のプリフィックスとして使用する場合は、「true」に設定します。watt.server.jms.nirvana.durableSubscriber.includeClientId が「true」に設定されていても、[**クライアントID**] プロパティが空の場合は継続的サブスクライバ名にはクライアント ID のプリフィックスが付加されません。クライアント ID をプリフィックスとして使用しない場合は、「false」に設定します。デフォルトは「true」です。

watt.server.jms.producer.pooledSession.timeout
Integration Server がエントリを削除する前に、非アクティブのエントリがデフォルトのセッションプールまたは宛先固有プールに保持される期間をミリ秒単位で指定します。Integration Server は、JMS 接続エイリアスの [**アイドルタイムアウト**] フィールドに -1 が指定されている場合に、この値を使用します。デフォルトは 60000 です。

watt.server.jms.trigger.caching.pollingInterval
プリフェッチキャッシュ機能 (コンシューマキャッシュ機能とも呼ばれる) を使用する JMS トリガーに対してメッセージを抽出するための Universal Messaging へのポーリング要求の間隔をミリ秒単位で指定します。デフォルトは 3000 ミリ秒です。

JMS トリガーに対するプリフェッチキャッシュ機能の設定方法の詳細については、*webMethods Service Development Help* または *Using webMethods Integration Server to Build a Client for JMS*を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.concurrent.consecutiveMessageThreshold
同時実行 JMS トリガーによる、メッセージに対する要求が連続して何回成功すると、そのトリガーがマルチスレッド処理されるようにするかを指定します。たとえば、このプロパティを 1000 に設定した場合、同時実行 JMS トリガーが JMS プロバイダから 1000 個の要求メッセージを連続して抽出すると、Integration Server によって、その JMS トリガーのマルチスレッド処理が開始されます。このプロパティのデフォルトは 0 であり、この場合、Integration Server では JMS トリガーがいつマルチスレッド処理されるようになるかを指定するしきい値を使用しません。しきい値を使用しない場合は、複数のメッセージを処理できるときには、常に、同時実行 JMS トリガーがマルチスレッド処理されます。同時実行 JMS トリガーに対してスレッド使用のしきい値を使用する方法の詳細については、920 ページの「同時実行 JMS トリガーで複数のスレッドを使用するためのしきい値の設定」を参照してください。

注:
同時実行 JMS トリガーにマルチスレッドを使用するためのしきい値は、Integration Server のすべての同時実行 JMS トリガーに適用されます。これには、標準 JMS トリガー、

SOAP-JMS トリガーおよび WS-エンドポイントトリガーが含まれます。しかし、しきい値は、Universal Messaging からメッセージを受け取り、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーには適用されません。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.concurrent.primaryThread.pollingInterval
同時実行 JMS トリガーのプライマリサーバスレッドが JMS プロバイダをポーリングしてメッセージを確認する間隔をミリ秒単位で指定します。JMS プロバイダからメッセージを抽出するサーバスレッドは、同時実行 JMS トリガーごとに用意されます。このスレッドが同時実行 JMS トリガーのプライマリスレッドと見なされます。ポーリング間隔が短い場合、Integration Server による CPU の使用量が増加し、JMS プロバイダの負荷が増大します。ただし、ポーリング間隔が短いと、負荷が高い場合や要求/応答シナリオでのパフォーマンスが向上する可能性があります。デフォルトは 500 ミリ秒です。

値 0 を指定した場合、JMS プロバイダからメッセージを抽出するために Integration Server によって javax.jms.MessageConsumer.receiveNoWait() API が使用されます。javax.jms.MessageConsumer.receiveNoWait() API は、すべての JMS プロバイダによってサポートされる場合とサポートされない場合があります。JMS トリガーによるメッセージの受信元の JMS プロバイダでこの API がサポートされない場合、JMS トリガーは無効になります。この場合は、watt.server.jms.trigger.concurrent.primaryThread.pollingInterval の値を正の整数に変更してから、JMS トリガーを有効にしてください。

注:
Universal Messaging に接続する場合、デフォルト値の 500 ミリ秒は最適ではありません。Universal Messaging が JMS プロバイダである場合、パフォーマンスを改善するために、Integration Server は 3000 ミリ秒の最小プライマリポーリング間隔を使用します。watt.server.jms.trigger.concurrent.primaryThread.pollingInterval プロパティが 3000 よりも大きい値に設定されている場合、Integration Server は Universal Messaging への接続に、このプロパティ値を使用します。

注:
JMS トリガーは複数の宛先からメッセージを抽出する必要があるため、比較的長いポーリング間隔は、結合を使用する JMS トリガーのパフォーマンスに影響を与えます。JMS トリガーがメッセージを抽出する宛先の 1 つが空である場合、JMS トリガーがこの宛先にメッセージを要求するたびに遅延が発生します。これは全体的なパフォーマンスに悪影響を与える可能性があります。

注:
watt.server.jms.trigger.concurrent.primaryThread.pollingInterval パラメータは、、Universal Messaging からメッセージを受け取り、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーには適用されません。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval

同時実行 JMS トリガーのセカンダリサーバスレッドが JMS プロバイダをポーリングしてメッセージを確認する間隔をミリ秒単位で指定します。セカンダリサーバスレッドは、同時実行 JMS トリガーのメッセージを抽出するために使用できる追加のスレッドです。同時実行 JMS トリガーのセカンダリサーバスレッド数は、[**実行スレッドの最大数**] プロパティの値から 1 を引いた値に対応します。ポーリング間隔が短い場合、Integration Server による CPU の使用量が増加し、JMS プロバイダの負荷が増大します。ポーリング間隔を長くすると、同時実行 JMS トリガーで使用される各セカンダリスレッドのポーリング間隔が経過するまで Integration Server が待機する必要があるため、Integration Server のシャットダウンに必要な時間に影響する可能性があります。デフォルト値は 1 ミリ秒です。

watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval の値は、watt.server.jms.trigger.concurrent.primaryThread.pollingInterval の値以下とする必要があります。

値 0 を指定した場合、JMS プロバイダからメッセージを抽出するために Integration Server によって javax.jms.MessageConsumer.receiveNoWait() API が使用されます。javax.jms.MessageConsumer.receiveNoWait() API は、すべての JMS プロバイダによってサポートされる場合とサポートされない場合があります。JMS トリガーによるメッセージの受信元の JMS プロバイダでこの API がサポートされない場合、予期しない動作が発生する可能性があります。この場合は、watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval の値を正の整数に変更します。

注:
Universal Messaging に接続する場合、デフォルト値の 1 ミリ秒は最適ではありません。Universal Messaging が JMS プロバイダである場合、パフォーマンスを改善するために、Integration Server は最小のセカンダリポーリング間隔を使用します。Universal Messaging からメッセージを受信する JMS トリガーが結合を使用する場合、Integration Server は 10 ミリ秒の最小セカンダリポーリング間隔を使用します。JMS トリガーが結合を使用しない場合、Integration Server は 3000 ミリ秒の最小セカンダリポーリング間隔を使用します。設定値が最小値よりも大きい場合、Integration Server は代わりに設定値を使用します。

注:
JMS トリガーは複数の宛先からメッセージを抽出する必要があるため、比較的長いポーリング間隔は、結合を使用する JMS トリガーのパフォーマンスに影響を与えます。JMS トリガーがメッセージを抽出する宛先の 1 つが空である場合、JMS トリガーがこの宛先にメッセージを要求するたびに遅延が発生します。これは全体的なパフォーマンスに悪影響を与える可能性があります。

注:
watt.server.jms.trigger.concurrent.secondaryThread.pollingInterval パラメータは、、Universal Messaging からメッセージを受け取り、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーには適用されません。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.extendedDelay.delayIncrementInterval
Integration Server が非アクティブの同時実行 JMS トリガーに対してポーリング遅延を使用する間隔をミリ秒単位で指定します。JMS トリガーが非アクティブのままである場合、この間隔により Integration Server が遅延時間を延長するタイミングが決まります。デフォルト値は

0 です。この値は、Integration Server が非アクティブな同時実行 JMS トリガーに対してポーリング遅延の延長を使用しないことを示します。ポーリング要求の遅延の詳細については、921 ページの「同時実行 JMS トリガーのポーリング要求の遅延」を参照してください。

注:
watt.server.jms.trigger.extendedDelay.delayIncrementInterval パラメータは、、Universal Messaging からメッセージを受け取り、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーには適用されません。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.extendedDelay.delays
非アクティブな同時実行 JMS トリガーが JMS プロバイダをポーリングしてメッセージを確認するまで待機する遅延時間を指定します。JMS トリガーが非アクティブ状態を維持している場合に遅延時間を増加するには、増加する整数値をカンマ区切りリストで指定します。watt.server.jms.trigger.extendedDelay.delays パラメータはミリ秒単位で指定し、デフォルト値は「0, 1000, 2000, 3000」です。

ポーリング要求の遅延の詳細については、921 ページの「同時実行 JMS トリガーのポーリング要求の遅延」を参照してください。

注:
watt.server.jms.trigger.extendedDelay パラメータは、、Universal Messaging からメッセージを受け取り、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーには適用されません。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.groupTag
トリガーグループに属する JMS トリガーの名前に使用されるグループタグを使用します。Integration Server では名前に指定されたグループタグを持つ JMS トリガーをトリガーグループのメンバーとして扱います。トリガーグループのすべてのトリガーで使用する名前の形式は folder.subfolder:originalJmsTriggerName_groupTag_Id です。

デフォルトは WMTG です。このパラメータを NULL (空白) に設定すると、Integration Server ではトリガーグループを使用しません。また、com.wm.app.b2b.server.jms.consumer.JMSTriggerGroupFacade クラスのメソッドを使用してトリガーを作成することはできません。

注:
watt.server.jms.trigger.groupTag パラメータの値を変更し、トリガーグループが既に Integration Server に存在する場合、以前のグループタグで作成されたトリガーはトリガーグループのメンバーとしては扱われません。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.maxDeliveryCount
JMS プロバイダがメッセージを Integration Server にデリバーできる最大回数を指定します。デフォルトは 100 です。

watt.server.jms.trigger.maxPrefetchSize

JMS トリガーに対する要求があるたびに webMethods Broker の JMS API で抽出およびキャッシュする最大メッセージ数を指定します。プリフェッチキャッシュを使用すると、webMethods Broker からのメッセージの抽出速度が上がります。メッセージは Integration Server のメモリに配置されるため、非常に大きなメッセージを受信する JMS トリガーを使用している場合は、必要に応じてこの設定を下げてください。そうしないと、メモリの問題が発生する可能性があります。このプロパティは、webMethods Broker からのメッセージを受信する JMS トリガーにのみ適用されます。デフォルトは 10 メッセージです。

Integration Server は、JMS トリガーが開始したときにこの値をチェックします。新しい値をすべての JMS トリガーに適用するには、Integration Server Administrator を使用して JMS トリガーを無効にしてから再度有効にする必要があります。Integration Server Administrator を使用して個々のトリガーを無効および有効にする方法の詳細については、911 ページの「JMS トリガーの管理」を参照してください。

このプロパティは、トリガーのプリフェッチプロパティが -1 に設定されている場合にのみ適用されます。

Integration Server のクラスタで作業している場合、最初はこのプロパティで制御される動作に関して誤解が生じる可能性があります。たとえば、2 つの Integration Server で構成されるクラスタが存在するとします。各 Integration Server では、同じ JMS トリガーが使用されています。20 個のメッセージが送信先に送信され、JMS トリガーはそこからメッセージを受信します。Integration Server 1 上の JMS トリガーが最初のメッセージを受信し、Integration Server 1 上の JMS トリガーが 2 番目のメッセージを受信し、それ以降も同様であると予想するかもしれません。ただし、実際には、Integration Server 1 上の JMS トリガーが最初の 10 個のメッセージを受信し、Integration Server 2 上の JMS トリガーが次の 10 個のメッセージを受信します。

注:
watt.server.jms.trigger.maxPrefetchSize parameter パラメータは、、Universal Messaging からメッセージを受け取り、プリフェッチキャッシュ機能を使用する同時実行 JMS トリガーには適用されません。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.jms.trigger.monitoringInterval

Integration Server が JMS トリガーに関するリソース監視サービスを実行する間隔を秒単位で指定します。リソース監視サービスは、JMS トリガーサービスによって使用されるリソースの使用可能性をチェックするためにユーザが作成するサービスです。再試行の試みがすべて失敗した場合、Integration Server は JMS トリガーを一時停止し、その後でシステムタスクをスケジュールして JMS トリガーに割り当てられているリソース監視サービスを実行します。デフォルトは 60 秒です。このプロパティの変更は、Integration Server が次回システムタスクをスケジュールして JMS トリガーに関するリソース監視サービスを実行するときに適用されます。

JMS トリガーに関するリソース監視サービスを構築する方法の詳細については、『*Using webMethods Integration Server to Build a Client for JMS*』を参照してください。

watt.server.jms.trigger.pollingSession.timeout

Integration Server が JMS トリガーの非アクティブな接続をリフレッシュする頻度 (分単位) を指定します。JMS トリガーが JMS トリガーセッションを使用してメッセージを抽出しない場合、このセッションは非アクティブと見なされます。デフォルトは 30 分です。

watt.server.jms.trigger.pooledConsumer.timeout
コンシューマプール内に保持されている非アクティブなコンシューマが削除されるまでの時間をミリ秒単位で指定します。Integration Server は、コンシューマのプールを使用して、同時実行 JMS トリガーに対するメッセージを抽出および処理します。タイムアウト値が経過すると、Integration Server はプール内の非アクティブコンシューマを削除します。Integration Server が同時実行 JMS トリガーに対するメッセージをさらに抽出できるようにするには、Integration Server は新しいコンシューマを作成する必要があります。その場合、javax.jms.MessageConsumer および javax.jms.Session を作成する必要があります。一部の JMS プロバイダでは、新しいコンシューマの作成は負荷の高い操作になる可能性があります。watt.server.jms.trigger.pooledConsumer.timeout サーバプロパティに大きい値を設定すると、JMS セッションおよび messageConsumer は長時間開いたままになります。Integration Server の負荷が高い場合にタイムアウト値を大きくすると、プールされたコンシューマの作成頻度が低くなります。一方、Integration Server の負荷が低い場合にタイムアウト値を大きくすると、未使用のプールされたコンシューマが存在する可能性があります。デフォルトは 20000 ミリ秒です。

watt.server.jms.trigger.raiseEventOnException
JMS トリガーでのメッセージの処理中に致命的な例外 (非一時的なエラーや再処理できないメッセージなど) が発生した場合に、Integration Server で JMS 抽出失敗イベントを生成するかどうかを指定します。Integration Server で JMS 抽出失敗イベントを生成する場合は、「true」を指定します。デフォルトは「true」です。

watt.server.jms.trigger.raiseEventOnRetryFailure
JMS トリガーで再試行失敗が発生した場合に、Integration Server で JMS 抽出失敗イベントを生成するかどうかを指定します。JMS プロバイダがメッセージの最大デリバー回数に達した場合、または ISRuntimeException がスローされたときに、メッセージを回復するように JMS トリガーが設定されていない場合 (たとえば、トランザクションに使用されていない JMS トリガーの再試行失敗処理が [一時停止して後で再試行] ではなく [例外をスロー] に設定されている場合)、再試行失敗が発生することがあります。「true」に設定した場合、Integration Server は再試行失敗の発生時に JMS トリガーの JMS 抽出失敗イベントを生成します。デフォルトは「true」です。

watt.server.jms.trigger.retryOnConsumerError
JMS プロバイダへの接続とは無関係なエラーにより JMS トリガーが無効化された場合、Integration Server が JMS トリガーの再有効化を自動的に試行するかどうかを決定します。watt.server.jms.trigger.retryOnConsumerError を「true」に設定すると、Integration Server は自動的に JMS トリガーを再有効化します。このプロパティを「false」に設定すると、メッセージコンシューマで JMS プロバイダへの接続とは無関係な予期しないエラーが発生した場合に、Integration Server は JMS トリガーを無効のままにします。デフォルトは「true」です。

watt.server.jms.trigger.reuseJmsTxSession
トランザクションに使用される JMS トリガーの複数のトランザクション間で、セッションを再利用できるかどうかを指定します。「true」に設定すると、トランザクションに使用される JMS トリガーは、JMS プロバイダからメッセージを受信するときに同じ JMS セッションを再利用します。「false」に設定すると、Integration Server は、トランザクションに使用される JMS トリガーが受信および処理するメッセージごとに、セッションを作成して閉じます。デフォルト値は true です。

一部の JMS プロバイダでは、複数のトランザクション間で単一のセッションを使用できない場合があります。このような JMS プロバイダを使用する場合は、watt.server.jms.trigger.reuseJmsTxSession を「false」に設定します。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.reuseSession
JMS トリガーの複数のインスタンスで Integration Server 上の同じセッションを使用するかどうかを指定します。このプロパティを「true」に設定した場合、JMS トリガーは共有セッションを使用します。JMS トリガーによって実行される各トリガーサービスが同じセッション ID を使用します。このプロパティを「false」に設定した場合、Integration Server は JMS トリガーのインスタンスごとに新しいセッションを使用します。JMS トリガーでセッションを再使用することにより、パフォーマンスが向上する可能性があります。ただし、このプロパティはすべてのアダプタに適用されるわけではありません。セッション ID を一意にする必要があるアダプタを使用している場合は、このプロパティを「false」に設定します。デフォルトは「false」です。

注:
Adapter for SAP を使用する場合、同時 JMS トリガーに pub.sap.client:lockSession および pub.sap.client:releaseSession サービスを呼び出すマルチステップ SAP トランザクションを実行するトリガーサービスがある場合は、watt.server.jms.trigger.reuseSession を「false」に設定します。これらのサービスには専用セッションが必要です。

watt.server.jms.trigger.serial.primaryThread.pollingInterval
逐次実行 JMS トリガーのサーバスレッドが JMS プロバイダをポーリングしてメッセージを確認する間隔をミリ秒単位で指定します。ポーリング間隔が短い場合、Integration Server による CPU の使用量が増加し、JMS プロバイダの負荷が増大します。ただし、ポーリング間隔が短いと、負荷が高い場合や要求/応答シナリオでのパフォーマンスが向上する可能性があります。デフォルトは 500 ミリ秒です。

値 0 を指定した場合、JMS プロバイダからメッセージを抽出するために Integration Server によって javax.jms.MessageConsumer.receiveNoWait() API が使用されます。javax.jms.MessageConsumer.receiveNoWait() API は、すべての JMS プロバイダによってサポートされる場合とサポートされない場合があります。JMS トリガーによるメッセージの受信元の JMS プロバイダでこの API がサポートされない場合、JMS トリガーは無効になります。この場合は、watt.server.jms.trigger.serial.primaryThread.pollingInterval の値を正の整数に変更してから、JMS トリガーを有効にしてください。

注:
Universal Messaging に接続する場合、デフォルト値の 500 ミリ秒は最適ではありません。Universal Messaging が JMS プロバイダである場合、パフォーマンスを改善するために、Integration Server は 3000 ミリ秒の最小プライマリポーリング間隔を使用します。トリガーが結合を使用する場合、Integration Server は 10 ミリ秒に設定されたセカンダリポーリング間隔を使用します。逐次実行 JMS トリガーに対して、セカンダリポーリング間隔を設定することはできません。

注:
JMS トリガーは複数の宛先からメッセージを抽出する必要があるため、比較的長いポーリング間隔は、結合を使用する JMS トリガーのパフォーマンスに影響を与えます。JMS トリガーがメッセージを抽出する宛先の 1 つが空である場合、JMS トリガーがこの宛先に

メッセージを要求するたびに遅延が発生します。これは全体的なパフォーマンスに悪影響を与える可能性があります。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.startupFailure.restartTaskRetryCount
JMS 接続エイリアスの起動時に開始に失敗した JMS トリガーを開始するために、トリガー再開タスクが実行する最大再試行回数を指定します。トリガーによって使用される JMS 接続エイリアスの起動時に、Integration Server は JMS トリガーに対して最初の開始を試行します。エイリアスの起動時に JMS トリガーが開始に失敗した場合、Integration Server はトリガーの再開始タスクを使用して失敗した JMS トリガーの開始を再試行します。再開始タスクは、エイリアスを開始するのに使用されたものとは別のスレッドを使用し、watt.server.jms.trigger.startupFailure.restartTaskRetryInterval パラメータによって指定された間隔で実行されます。再開始タスクを使用するには、1 以上の正の整数を指定する必要があります。デフォルトの再試行回数は 6 回です。Integration Server が JMS 接続エイリアスの起動時に開始に失敗した JMS トリガーを再開始させないようにするには、watt.server.jms.trigger.startupFailure.restartTaskRestryCount を 0 に設定します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.startupFailure.restartTaskRetryInterval
JMS 接続エイリアスの起動時に開始に失敗した JMS トリガーを再開するために、トリガーの再開タスクが試行間に待機する秒数を指定します。正の整数を指定する必要があります。デフォルトは 30 秒です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.startupFailure.retryCount
トリガーの開始が失敗した後に Integration Server が JMS トリガーの開始を再試行する回数の最大値を示します。トリガーの最初の開始の失敗後、Integration Server は 1000 ミリ秒待機してから、トリガーの開始を再試行します。トリガーの開始が失敗すると、Integration Server では 1000 ミリ秒待機してから、トリガーの開始を再試行します。JMS トリガーが正常に開始されるか、JMS 接続エイリアスの実行が停止するか、または JMS トリガーが無効になるまで、Integration Server ではこのような処理が継続されます。Integration Server が最大回数の再試行を行った後にトリガーの開始が失敗した場合、Integration Server ではそれ以上トリガーを開始するアクションを実行できません。Integration Server では例外がログに記録され、Integration Server Administrator の [JMS トリガーの管理] ページに表示されます。問題が解決され、トリガーが手動で再び開始 (つまり有効に) されるまで 、JMS トリガーは非アクティブな状態のままです。

watt.server.jms.trigger.startupFailure.retryCount パラメータは 0 以上の整数に設定する必要があります。パラメータを 0 に設定すると (デフォルト)、Integration Server では再試行を行いません。再試行回数が多すぎると、JMS 接続エイリアスの開始時間が遅れます。

重要:
このパラメータの設定を変更した場合、変更を適用するには、再起動する必要があります。

watt.server.jms.trigger.stopRequestTimeout
JMS トリガーが無効になってから JMS トリガーによるメッセージの処理を強制的に停止するまでの Integration Server の最大待機時間を秒単位で指定します。最小値は 0 です。デフォルトは 3 秒です。

注:
watt.server.jms.trigger.stopRequestTimeout を 0 に設定した場合、JMS トリガーが無効になると、JMS トリガーは即座にメッセージの処理を停止します。この場合、Integration Server はトリガーを即座に停止し、処理が完了するまで待機しません。これにより、javax.jms.IllegalStateException が生成される可能性があります。

JMS トリガーを無効にするには、pub.trigger:disableJMSTrigger サービスを呼び出すか、Integration Server Administrator の [**メッセージング**] > [JMS トリガー] ページを使用します。JMS トリガーを無効にする方法の詳細については、913 ページの「JMS トリガーのステータスと状態について」を参照してください。

Designer で JMS トリガーを保存すると、Integration Server はトリガーを停止してから再開します。この状況では、Integration Server は 3 秒間待機してから JMS トリガーによるメッセージの処理を強制的に停止します。この値は設定できません。

注:
これは、メッセージを処理している間で、JMS トリガーが無効になっている場合です。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.trigger.wmjms.clientIDSharing
Integration Server で継続的サブスクライバーからのメッセージを負荷分散方式で受信するかどうかを指定します。「true」に設定した場合、Integration Server は次のように動作します。

- JMS トリガーで指定された継続的サブスクライバが存在しない場合、Integration Server は Broker で継続的サブスクライバを作成し、状態共有を有効にします。また、Integration Server は、メッセージ処理モードを使用して、継続的サブスクライバに対して状態共有での順序付けモードを設定します(逐次実行のメッセージ処理モードは状態共有での順序付けモードの [パブリッシャー] にマッピングされ、同時実行のメッセージ処理モードは状態共有での順序付けモードの [None] にマッピングされます)。

  継続的サブスクライバが既に存在する場合、Integration Server は JMS トリガーの保存時に継続的サブスクライバを変更しません。

- Integration Server 内の com.webmethods.jms.clientIDSharing プロパティを「true」に設定することによってクライアント識別子を共有できることを Broker に示します。

このプロパティは、JMS トリガーで使用される JMS 接続エイリアスがネイティブ webMethods API を使用して webMethods Broker に接続する場合にのみ適用されます。

デフォルトは「`true`」です。

注:

webMethods Broker が廃止されたため、このパラメータは廃止されました。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.jms.wmjms.lms.readTimeout
Integration Server が入力ストリームの次の部分を受信するまでに待つ時間 (ミリ秒単位) を指定します。この時間を過ぎると、WmReadTimeoutException がスローされます。読み取りタイムアウトは、Integration Server が入力ストリームの最初の部分を取得した時点から適用されます。デフォルトは 30000 ミリ秒です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.jndi.searchresult.maxlimit
LDAP またはセントラルディレクトリでユーザまたはグループを検索するときに検索条件を指定しなかった場合に、Integration Server に表示されるレコードの最大数を指定します。デフォルトは `0` です。0 に設定した場合、Integration Server には使用可能なすべてのレコードが表示されます。

watt.server.jndi.searchresult.pageSize
Integration Server が、接続された LDAP サーバから抽出するユーザグループのバッチサイズを指定します。このパラメータを使用して、LDAP 検索結果のページサイズを制限します。LDAP サーバからすべてのユーザグループを一度に抽出するには、このパラメータを 0 に設定します。デフォルト値は 1000 です。

watt.server.json.allowUnquotedFieldNames
JSON 標準では、フィールド名を二重引用符で囲む必要があります。ただし、JavaScript ではフィールド名を二重引用符で囲む必要がないため、レガシー JavaScript を JSON テキストとして解析するときは、引用符なしのフィールド名が許可されると便利な場合があります。このパラメータを「`true`」に設定した場合、pub.json:jsonStringToDocument サービスおよび pub.json:jsonStreamToDocument サービスにおいて、指定された JSON テキスト内で引用符なしのフィールド名が許可されます。このパラメータを「`false`」に設定した場合、引用符なしのフィールド名が出現するとサービスは ServiceException をスローします。デフォルトは「`false`」です。

watt.server.json.allowUnquotedFieldNames パラメータは、JSON テキストからドキュメントタイプを作成するプロセスにも影響を及ぼします。watt.server.json.allowUnquotedFieldNames を「`true`」に設定した場合、引用符なしのフィールドを含む JSON テキストがドキュメントタイプのソースとして使用されると、作成されたドキュメントタイプには、引用符なしのフィールドおよび引用符付きのフィールドに対応するフィールドが含まれます。
watt.server.json.allowUnquotedFieldNames を「`false`」に設定した場合、引用符なしのフィールドを含む JSON テキストがドキュメントタイプのソースとして使用されると、Designer は例外をスローし、ドキュメントタイプを作成しません。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.json.decode.unescapeSpecialChars

Integration Server が、JSON ドキュメントを解析するときに、特殊文字 '\n'、'\r'、'\t'、'\b'、'\f'、'\\'、'\" をエスケープするかどうかを指定します。このパラメータが「true」に設定されている場合、Integration Server は、出力ドキュメント内でこれらの特殊文字をエスケープしません (つまり、'\n' は新しい行に置き換えられます。他の文字も同じように置き換えられます）。このパラメータが「false」に設定されている場合は、Integration Server は、出力ドキュメント内でこれらの文字をそのままに保持します。このサーバプロパティのデフォルト値は「true」です。

watt.server.json.decodeIntegerAsLong
Integration Server が JSON コンテンツから抽出した整数を Java ラッパータイプの Long または Integer に変換します。このパラメータを「`true`」に設定した場合、Integration Server は JSON 整数を Java ラッパータイプの Long に変換します。このパラメータを「`false`」に設定した場合、Integration Server は JSON 整数を Java ラッパータイプの Integer に変換します。デフォルトは「true」 (Long) です。

watt.server.json.decodeNullRootAsEmpty
Integration Server は、JSON コンテンツから抽出した NULL 値を IData または空の IData に変換します。このパラメータを「`false`」に設定した場合、Integration Server は NULL 値を IData に変換します。このパラメータを「`true,`」に設定した場合、Integration Server は、NULL 値を空の IData に変換し、空の IData のエンコーディングが後で "{}" の JSON テキストを作成します。この JSON コンテンツは、元の NULL 値が "{}" の JSON テキストに変換されるため、元の JSON コンテンツ (NULL) とは違うものになります。デフォルトは「`false`」です。

watt.server.json.decodeRealAsDouble
Integration Server が JSON コンテンツから抽出した実数を Java ラッパータイプの Float または Double に変換します。このパラメータを「true」に設定した場合、Integration Server は実数を Java ラッパータイプの Double に変換します。このパラメータを「`false`」に設定した場合、Integration Server は実数を Java ラッパータイプの Float に変換します。デフォルトは「`true`」 (Double) です。

watt.server.json.decodeRealAsString
Integration Server が JSON コンテンツから取得する実数を String に変換します。このパラメータを「`true`」に設定した場合、Integration Server は実数を String に変換します。このパラメータを「`false`」に設定した場合、Integration Server は実数を String に変換しません。代わりに、*decodeRealAsDouble* に指定された値に従って、実数は Java ラッパータイプの Float または Double に変換されます。デフォルトは「`false`」です。

> **注:**
> watt.server.json.decodeRealAsString および watt.server.json.decodeRealAsDouble の両方が「`true`」に設定された場合、エラーが発生します。

watt.server.json.encodeDateAs
java.util.Date オブジェクトが、IDataJSONCoder.encode() と pub.json:documentToJSONString サービスを含む Integration Server によりどのように形式変換されるのかを指定します。JSON コンテンツを含むクライアント要求に Integration Server が応答する場合、その応答内の java.util.Dates の形式はこのパラメータによって制御されます。

- 日付をタイムスタンプ（1970 年 1 月 1 日 00:00:00 以降のミリ秒数）にエンコードするには、このパラメータを「long」に設定します。日付は JSON 数値としてエンコードされます。
- java.util.Date インスタンスを標準の ISO 形式である `YYYY-MM-DD'T'HH:mm:ss.sssZ` にエンコードするには、このパラメータを「ISO8601」に設定します。日付は JSON ストリングとしてエンコードされます。
- java.util.Date のインスタンスを指定したパターンにエンコードするには、このパラメータをカスタム形式に設定します。指定するパターンは、Oracle Java API マニュアルの java.text.SimpleDateFormat クラスで説明されている日付と時間のパターンに準拠する必要があります。日付は JSON ストリングとしてエンコードされます。

watt.server.json.encodeDateAs が設定されていない場合は、java.util.Date インスタンスは標準 ISO 形式である `EEE MMM dd hh:mm:ss zzz yyyy` にエンコードされます。このデフォルトの形式は、java.util.Date.toString() によって提供されています。

サービスの呼び出しに使用する URL に `jsonDateEncoding=`*format* (*format* は上記のいずれか 1 つ）のクエリーパラメータを含めると、invoke、rest、restv2、または admin ディレクティブを使用する個別の要求でこの設定を上書きできます。pub.json:documentToJSONString サービスを *encodeDateAs* 入力パラメータとともに呼び出すことで、設定を上書きすることもできます。pub.json:documentToJSONString の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

注:
soap または ws など、他のディレクティブを使用している場合はこの設定を上書きできません。

watt.server.json.iterator.maxBatchSize

`pub.json:getNextBatch` サービスが抽出できる配列エレメントの最大数を指定します。このプロパティの値は、`watt.server.json.iterator.minBatchSize` の値以上である必要があります。それ以外の場合は、このプロパティのデフォルト値が使用されます。

`watt.server.json.iterator.minBatchSize` の値が 0 (デフォルト) の場合、このプロパティには任意の正の整数を設定できます。

`pub.json:getNextBatch` サービスの `batchSize` パラメータはこのプロパティの値以下である必要があります。それ以外の場合は、サービスはエラーをスローします。デフォルト値は 0 で、バッチサイズに最大値制限がないことを示します。

watt.server.json.iterator.maxConcurrentRequests

Integration Server が処理する JSON 反復子の同時実行要求の最大数を指定します。このプロパティに正の整数が指定されていた場合、Integration Server は同時実行反復子を設定に応じて制限します。これにより、要求の応答時間が長くなることがあります。しかし、この設定によりメモリ不足の問題を避けることができます。デフォルト値は 0 で、JSON 反復子要求の同時実行に最大制限がないことを示します。

注:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.json.iterator.minBatchSize

`pub.json:getNextBatch` サービスが抽出できる配列エレメントの最小数を指定します。このプロパティの値は、`watt.server.json.iterator.maxBatchSize` の値以下である必要があります。

それ以外の場合は、このプロパティのデフォルト値が使用されます。
watt.server.json.iterator.maxBatchSize の値が 0 (デフォルト) の場合、このプロパティには任意の正の整数を設定できます。

pub.json:getNextBatch サービスの batchSize パラメータはこのプロパティの値以上である必要があります。それ以外の場合は、サービスをエラーをスローします。デフォルト値は0で、バッチサイズに最小値制限がないことを示します。

watt.server.json.optimizeForUniqueKeys
ドキュメントのキーが一意であるか重複しているかに応じて、Integration Server が JSON ドキュメントを解析する方法を制御します。受信するドキュメントの特性に応じてこのパラメータを設定すると、Integration Server は JSON ドキュメントをより迅速に解析できます。一意のキーまたは少数の重複キーがある JSON ドキュメントを処理する場合、このプロパティを「true」(デフォルト) に設定します。多数の重複キーがある JSON ドキュメントを処理する場合、このプロパティを「false」に設定します。

注:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.json.prettyPrint
出力を読みやすくするために、復帰 (CR) とインデントを使用して JSONCoder.encode() の出力を書式設定するかどうかを指定します。JSON コンテンツを含むクライアント要求に Integration Server が応答する場合、その応答の形式はこのプロパティによって制御されます。このパラメータを「true」に設定した場合、Integration Server は復帰 (CR) およびインデントを使用して JSON をエンコードし、読みやすくします。デフォルトは「false」です。
JSONCoder の詳細については、『*webMethods Integration Server Java API Reference*』を参照してください。
サービスの呼び出しに使用する URL に jsonPrettyPrint=true または jsonPrettyPrint=false のクエリーパラメータを含めると、invoke、admin、rest、または restv2 ディレクティブを使用する個別の要求でこの設定を上書きできます。

注:
soap または ws など、他のディレクティブを使用している場合はこの設定を上書きできません。

watt.server.json.prettyPrint を設定すると、pub.json:documentToJSONString サービスで *jsonString* 出力パラメータのプリティプリントを使用するかどうかも指定できます。
pub.json:documentToJSONString の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.json.quoteFieldNames
pub.json:documentToJSONString サービスで生成されるすべての JSON フィールド名を二重引用符で囲むかどうかを指定します。JSON 標準では、フィールド名を二重引用符で囲む必要があります。ただし、生成される JSON テキストが旧式の JavaScript インタプリタで処理される場合は、必要に応じて、このサービスで引用符なしのフィールド名を生成してください。このパラメータを「true」に設定した場合、pub.json:documentToJSONString サービスでは、出力される JSON テキストのフィールド名は二重引用符で囲まれます。このパラメータを「false」に設定した場合、生成される JSON テキストのフィールド名は二重引用符で囲まれません。デフォルトは「true」です。

注:

watt.server.json.quoteFieldNames を「false」に設定するときは注意してください。この場合は pub.json:documentToJSONString サービスは標準でない JSON テキストを生成します。このため、JSON テキストがその他の組織に送信されるときに、相互運用性の問題が発生する可能性があります。

注:
変更内容を有効にするには、Integration Server を再起動する必要があります。

watt.server.key
サーバのライセンスキーを指定します。デフォルトはありません。

watt.server.ldap.checkLocalUserInLDAP
Integration Server が、接続された LDAP サーバによりローカルユーザを認証するかどうかを指定します。ユーザ名がローカルシステムに存在する場合でも、Integration Server が、接続された LDAP サーバによりローカルユーザを認証するには、このパラメータを「`true`」に設定します。ユーザ名がローカルシステムに存在する場合に LDAP ベースの認証をスキップするには、このパラメータを「`false`」に設定します。このパラメータのデフォルト値は「`false`」です。

watt.server.ldap.cleanContext
pub.client.ldap サービス要求を処理した後で、Integration Server が TCP/IP 接続を閉じて LDAP コンテキスト (connectionHandle オブジェクト) を消去する必要があるかどうかを指定します。このプロパティを「`true`」に設定した場合、Integration Server は接続を閉じて基礎的な LDAP コンテキストを消去します。Integration Server は LDAP コンテキストをパイプラインに返しません。このプロパティを「`false`」(デフォルト) に設定した場合、Integration Server は接続を閉じずに LDAP コンテキストをパイプラインに返します。

pub.client.ldap サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.ldap.DNescapeChars
LDAP サーバに提示される識別名の中で Integration Server が (\ 文字を使用して) エスケープできる文字を指定します。デフォルトでは、このプロパティは NULL に設定されています。つまり、文字はエスケープされません。

使用する識別名に Integration Server でのエスケープが必要な特殊文字が含まれている場合、このパラメータを使用します。たとえば、LDAP サーバで「`abc/def`」などのユーザ ID が保持される場合、次のように指定します。

```
watt.server.ldap.DNescapeChars = /
```

\ 文字をエスケープする必要がある場合、次のように \ を 2 回指定します。

```
watt.server.ldap.DNescapeChars = \\
```

watt.server.ldap.DNescapePairs
LDAP 識別名の中で、LDAP サーバに提示される前に Integration Server が二重にエスケープしてはいけない文字を指定します。識別名の中の特殊文字をエスケープするには、その文字の前にバックスラッシュ (\) を付けます。デフォルトでは以下の文字が二重エスケープされないため、watt.server.ldap.DNescapePairs の中で指定しないでください。

, = + < > # ; / \

LDAP サーバの識別名にこれら以外の特殊文字が含まれる場合は、watt.server.ldap.DNescapePairs に追加して、Integration Server がその文字を二重エスケープしないようにします。たとえば、等号 (=) はデフォルトで二重エスケープされないため、watt.server.ldap.DNescapePairs に追加しないでください。ただし、LDAP 識別名にアンパサンド (&) が含まれるため、二重エスケープされないようにするには、以下のようにアンパサンド文字を watt.server.ldap.DNescapePairs に追加します。

```
watt.server.ldap.DNescapePairs=&
```

このプロパティにはデフォルト値がありません。値が指定されない場合、前述以外の特殊文字は二重エスケープの対象となりません。このプロパティは動的ではありません。この値に対する変更は、Integration Server を再起動すると有効になります。

watt.server.ldap.DNescapeURL
ドメイン名の先頭にある 3 つのスラッシュ (/) を Integration Server が無視するのか、エスケープ文字として扱うのかを示します。「`false`」 (デフォルト) に設定すると、Integration Server はドメイン名の先頭にある 3 つのスラッシュを無視します。Integration Server は、watt.server.ldap.DNescapeChars および watt.server.ldap.DNescapePairs サーバ設定パラメータで定義されたエスケープ文字に関係なく、エスケープ文字としてではなくドメイン名の一部としてスラッシュを扱います。

「`true`」に設定すると、Integration Server はエスケープ文字としてスラッシュを扱い、watt.server.ldap.DNescapeChars および watt.server.ldap.DNescapePairs の設定に従ってドメイン名をエスケープします。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.server.ldap.DNstripQuotes
Integration Server が LDAP サーバに提示されるドメイン名から先頭および末尾の引用符を取り除くかどうかを指定します。「`true`」 (デフォルト) に設定すると、Integration Server は特殊文字が含まれるドメイン名から引用符を取り除きます。

注:
watt.server.ldap.DNstripQuotes を「`false`」に設定すると、Integration Server はドメイン名に含まれる引用符をそのまま残します。このため、ルックアップが停止することがあります。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.server.ldap.doNotBind
Integration Server が LDAP サーバに認証されるかどうかを指定します。Integration Server がカスタム認証モジュールを使用しており、ユーザが LDAP ディレクトリに認証される必要がない場合は、このプロパティを「`true`」に設定して、不要な要求が LDAP サーバに送信されないようにします。デフォルトは「`false`」です。

watt.server.ldap.extendedMessages
Integration Server が、認証エラーが発生したときに LDAP サーバから返された追加情報を表示するかどうかを指定します。この情報を使用できるのは、LDAP サーバが情報を提供している場合だけです。Active Directory は、この追加情報を提供する LDAP サーバです。デフォルトは「`false`」です。

「false」に設定すると、エラーメッセージは次のように表示されます。

```
2005-03-08 15:40:33 EST [ISS.0002.0035E]
CN=bob,OU=ISUsers,DC=KQA,DC=WEBMETHODS,DC=COM として
ldap://10.3.33.203:389/dc=KQA,dc=webMethods,dc=com に接続中に無効なクレデンシャルが検知されました。
```

「true」に設定されていると、このエラーは次のように表示されます。

```
2005-03-08 15:40:33 EST [ISS.0002.0035E]
CN=bob,OU=ISUsers,DC=KQA,DC=WEBMETHODS,DC=COM
として ldap://10.3.33.203:389/dc=KQA,dc=webMethods,dc=com
に接続中に無効なクレデンシャルが検知されました:
[LDAP: error code 49 - 80090308: LdapErr: DSID-0C09030F, comment:
AcceptSecurityContext error, data 52e, vece]
```

Active Directory ユーザの場合、データコード (上記の data 52e) には、認証に失敗した理由が示されます。このコードを 10 進数に変換し、Microsoft Web サイト (http://msdn.microsoft.com/library/en-us/debug/base/system_error_codes.asp) で検索して、問題を特定することができます。

watt.server.ldap.extendedProps
JNDI コンテキストを初期化するときに Integration Server が LDAP 実装に直接渡す LDAP 環境プロパティを指定します。この形式を以下に示します。

たとえば、デフォルトではない専用の JNDI プロバイダを使用しているか、コンテキストが作成されたときに環境に渡される特殊な JNDI プロパティが LDAP ディレクトリに必要な場合、以下のようにプロパティ customProperty を customValue に設定できます。

```
watt.server.ldap.extendedProps=java.naming.customProperty=customValue
```

デフォルトはありません。

watt.server.ldap.groupSearchFilter
Integration Server および SPM (SoftwareAG Platform Manager) プラグインが、接続された LDAP サーバでユーザグループを検索するときに使用するフィルタを指定します。このパラメータは検索結果の絞り込みに役立ちます。デフォルト値はありません。

watt.server.ldap.includeOnlyActiveGroups
空のグループを ACL に関連付けられるかどうかを指定します。1 メンバー以上の LDAP グループのみ ACL に関連付けられるようにする場合、「true」に設定します。空のグループを含め、すべての LDAP グループを ACL に関連付けられるようにする場合、「false」に設定します。デフォルトは「true」です。

> **注:**
> Integration Server Administrator では、ACL に割り当てるグループを検索するため、検索条件を入力できます。watt.server.ldap.includeOnlyActiveGroups を「false」に設定すると、検索結果には条件に一致するすべての LDAP 要素 (グループ以外を含む) が返されます。

watt.server.ldap.memberInfoInGroups
Integration Server が LDAP グループのメンバーシップ情報をどこで検索するかを制御します。これをデフォルトの「true」に設定すると、Integration Server はグループオブジェクトからグループのメンバーシップ情報を検索します。これを「false」に設定すると、Integration Server はユーザオブジェクトからグループのメンバーシップ情報を検索します。

watt.server.ldap.retryCount
ネットワーク障害または LDAP サーバの再起動の後に、Integration Server が LDAP サーバに自動的に再接続を試行する回数を指定します。0 に設定した場合、Integration Server は接続を再試行せず、LDAP ユーザにクレデンシャルの提示を求めます。これを正の整数に設定すると、Integration Server は指定された回数だけ接続を再試行します。デフォルトは 5 です。

watt.server.ldap.retryWait
ネットワーク障害または LDAP サーバの再起動の後に、Integration Server が LDAP サーバに再接続を試行する前に待機する時間をミリ秒単位で指定します。0 に設定した場合に、LDAP との通信で一時的な障害が発生すると、Integration Server は LDAP サーバへの再接続を行いません。正の整数に設定すると、Integration Server は `watt.server.ldap.retryCount` に指定された回数だけ接続を再試行し、`watt.server.ldap.retryWait` に指定された時間待機してから次の回の再試行を試みます。デフォルトは 2000 (ミリ秒) です。

watt.server.licenses
ライセンス数を指定します。デフォルトは 1 です。

watt.server.log.alertMaxEntries
[ログ] > 「logName」ページに表示できるエントリ数を指定します。この数値を超えると、Integration Server Administrator はアラートメッセージを表示します。**[ログ] > 「logName」**ページの **[表示するエントリ数]** フィールドの入力値が、watt.server.log.alertMaxEntries に指定された値よりも大きくなっている場合、Integration Server Administrator によりメッセージが表示され、要求されたエントリ数がしきい値を超えており、さらに多くのエントリを表示しようとすると、Integration Server のパフォーマンスに影響が及ぶ可能性があると警告されます。watt.server.log.alertMaxEntries に値が指定されていない場合、Integration Server Administrator はアラートメッセージを表示しません。

watt.server.log.maxEntries
ログ表示ユーティリティに表示するログエントリ数のデフォルト値を指定します。デフォルトは 35 エントリです (最新のエントリから)。詳細については、246 ページの「すべてのログの表示を永続的に変更」を参照してください。

watt.server.log.orphanLoggers
IS ログ階層で採用する孤立したロガーを指定します。デバッグより高いログレベルが設定されていても、Integration Server のログがデバッグメッセージでいっぱいになる場合があります。これは、Integration Server に孤立したロガー (IS ログツリー階層の一部ではないロガー) があり、カスタムアプリケーションで log4j のルートロガーが設定されている場合に発生する可能性があります。log4j のカスタム設定がない場合、孤立したロガーは IS ログツリー階層で指定されたログレベルを継承します。ただし、カスタムアプリケーションで log4j 設定ファイルのルートロガーを再設定する場合、孤立したロガーはルートロガーのログレベルを継承します。このため、log4j.rootLogger が DEBUG に設定されている場合、孤立したロガーは Debug レベルでログを記録します。孤立したロガーが変更された log4j.rootLogger レベルでログを記録しないようにする場合は、このパラメータを使用してください。複数の孤立したロガーを区切る場合はカンマを使用します。次に例を示します。

```
watt.server.log.orphanLoggers=WEBMETHODS.DEFAULT,
log4j.logger.com.softwareag.wsstack,log4j.logger.com.softwareag.security,
Debug.com.webmethods.lwq,com.webmethods.portal.jms.db.DbJMSClient
```

これがデフォルト値です。

watt.server.log.queued

サーバの機能によってメモリに書き込まれたログエントリをキューに入れ、バックグラウンドスレッドを使用してそれらのログエントリをサーバログに書き込むかどうかを指定します。デフォルトは「true」(ログエントリをキューに入れる) です。詳細については、『*webMethods Audit Logging Guide*』を参照してください。

watt.server.log.refreshInterval
表示されたサーバログを Integration Server Administrator がリフレッシュする時間を秒単位で指定します。デフォルトは 90 秒です。詳細については、246 ページの「すべてのログの表示を永続的に変更」を参照してください。

watt.server.logEncoding
Integration Server がログファイルまたはコンソールとのテキストファイルの読み取りと書き込みに使用するエンコーディングを指定します。デフォルトは「UTF-8」です。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.logRotateInterval
ログエントリのログ再利用間隔の長さ (ミリ秒単位) を指定します。Integration Server 9.7 以降、このサーバ設定パラメータは名前が watt.server.statsLogRotateInterval に変更されています。

8.2 SP2 以降、watt.server.logRotateInterval パラメータは Integration Server から削除されました。以下の修正で再度実装された際に、統計ログのみに影響するように、パラメータの範囲が変更されました。

- IS_9.0_SP1_Core_Fix6
- IS_9.5_SP1_Core_Fix3
- IS_9.6_Core_Fix2

Integration Server 9.6 またはそれ以前のバージョンから Integration Server 9.7 以上に移行した場合、Integration Server では次のことが実行されます。

- server.cnf 内のサーバ設定パラメータの名前が watt.server.logRotateInterval から watt.server.statsLogRotateInterval に変更されます。
- サーバ設定パラメータの値がミリ秒から分に変換されます。
- watt.server.statsLogRotateInterval を使用して、統計ログファイルのログ再利用間隔が設定されます。

watt.server.loginFailureLimit
ユーザが誤ったログインクレデンシャルを入力できる回数を指定します。この回数を超過すると、Integration Server は管理者にアラートを送信し、ログイン失敗カウントを 0 にリセットします。たとえば、watt.server.loginFailureLimit を 5 に設定した場合、ユーザが誤ったログインクレデンシャルを 6 回入力すると、Integration Server は電子メールアラートを管理者に送信し、ログイン失敗カウントを 0 にリセットします。デフォルト値は 5 です。

watt.server.login.userFtpDir
FTP リスナーを介して Integration Server に接続しているユーザをデフォルトの FTP ルートディレクトリとクライアントユーザディレクトリのどちらに配置するかを指定します。この

プロパティを「false」に設定した場合 (または指定しなかった場合)、ユーザは FTP ルートディレクトリにログインします。その後、ユーザは cd コマンドを発行してクライアントユーザディレクトリにアクセスする必要があります。このプロパティを「true」に設定した場合、ユーザはクライアントユーザディレクトリにログインします。デフォルト値は「false」です。

FTP ルートディレクトリは、Integration Server のホームディレクトリ内の「userFtpRoot」という名前のデフォルトディレクトリか、ユーザが watt.server.userFtpRootDir プロパティで定義したディレクトリになります。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.server.logs.dateStampFmt
監査ログファイル (FailedAudit_*、WMERROR*、WMSESSION*、WMTXIN*、WMTXOUT*) で使用する日付とタイムスタンプの形式を指定します。指定する形式は、java.text.SimpleDateFormat クラスに準拠している必要があります。

watt.server.logs.dateStampFmt プロパティを設定しなかった場合、Integration Server はデフォルト形式の *yyyy-MM-dd*T*hh:mm:ss.SSS*Z (たとえば、2010-04-19T19:07:21.505Z) を使用します。

watt.server.logs.dateStampTimeZone
監査ログファイル (FailedAudit_*、WMERROR*、WMSESSION*、WMTXIN*、WMTXOUT*) で使用するタイムゾーンを指定します。指定する形式は、java.util.TimeZone クラスに準拠している必要があります (たとえば、watt.server.logs.dateStampTimeZone=EST)。このプロパティを設定しなかった場合、Integration Server はホストのローカルタイムゾーンを使用します。このプロパティを適用するには、watt.server.logs.dateStampFmt の指定も必要です。

watt.server.math.floatOperation.mode
pub.math:*Floats サービスが、2 つの浮動小数点数を使用する演算の正確な結果、JVM によって計算された結果、固定の小数点以下桁数に基づく結果のどれを返すかを指定します。

このパラメータは、次のいずれかの値に設定します。

指定内容	pub.math:*Floats services to
dynamic	正確な演算結果を返します。たとえば、pub.math:addFloats を使用して 62.98 と 23 を加算した場合、サービスは合計として 85.98 を返します。
default	JVM によって計算された演算結果を返します。 たとえば、pub.math:addFloats を使用して 62.98 と 23 を加算した場合、サービスは合計として 85.97999999999999 を返します。 これがデフォルトの動作です。
fixed	あらかじめ決められた小数点以下桁数で四捨五入された結果を返します。pub.math:addFloats および pub.math:subtractFloats サービスの場合、Integration Server は結果を小数点以下 3 桁で四捨五入します。 pub.math:multiplyFloats サービスの場合、Integration Server は積を小数点以下 16 桁で四捨五入します。

指定内容	pub.math:*Floats services to pub.math:divideFloats サービスの場合、Integration Server は結果を小数点以下 17 桁で四捨五入します。

注:
pub.math:*Floats サービスの *precision* 入力パラメータの値を指定した場合、*precision* の値が 'watt.server.math.floatOperation.mode プロパティの値よりも優先されます。
pub.math:*Floats サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.mediator.directives
API Gateway 要求の検出に使用されるディレクティブのカンマ区切りリストを指定します。Integration Server が watt.server.mediator.directives サーバ設定パラメータで指定されたディレクティブを使用する要求を受信すると、Integration Server はこの要求によって呼び出された Web サービスが API Gateway サービスであるかどうかをチェックします。Web サービスが API Gateway サービスである場合、Integration Server は要求にデフォルトユーザを割り当てます。API Gateway のデフォルトディレクティブは「ws,mediator,gateway」です。

watt.server.messaging.csq.maxRedeliveryCount
クライアントサイドキュー (CSQ) から Universal Messaging にメッセージをパブリッシュするときに Integration Server が実行する最大再デリバー試行回数を指定します。Integration Server がメッセージをパブリッシュし、Universal Messaging が使用可能でないときに、Universal Messaging エイリアスと共に CSQ を使用するように設定されている場合、Integration Server は保証付きメッセージを CSQ に書き込みます。Universal Messaging が使用可能になると、Integration Server は、CSQ から Universal Messaging へメッセージをパブリッシュすることによって CSQ を排出します。CSQ から Universal Messaging へメッセージをパブリッシュする最初の試行が失敗した場合、Integration Server はメッセージが正常にパブリッシュされるまで、またはこのパラメータで指定された最大回数まで再試行を実行します。CSQ から Universal Messaging にパブリッシュする各試行は再デリバー試行と見なされます。Integration Server が最大回数の再デリバー試行を行い、すべての試行が失敗した場合、Integration Server はドキュメントを破棄します。watt.server.messaging.csq.maxDeliveryCount パラメータには正の整数を設定する必要があります。デフォルト値は 5 です。

注:
このパラメータは、Universal Messaging がメッセージングプロバイダである webMethods メッセージングに適用されます。Broker がメッセージングプロバイダである webMethods メッセージングで使用するプロパティは watt.server.publish.maxCSQRedeliveryCount です。プロバイダに関係なく、JMS メッセージングで使用するパラメータは watt.server.jms.csq.maxRedeliveryCount です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.messaging.csq.publishDelayWhileDraining

Integration Server がクライアントサイドキュー (CSQ) にメッセージをパブリッシュする前に CSQ の排出中に待機する秒数を指定します。CSQ に 1 つ以上のメッセージが含まれているときに遅延を使用するには、0 より大きく 60 以下の整数を指定します。

CSQ 内のメッセージ数によって遅延の長さを変化させるよう Integration Server に指示するには、0 を指定します。watt.server.messaging.csq.publishDelayWhileDraining を 0 に設定した場合、Integration Server の待機時間は次のとおりです。

- CSQ のメッセージ数が 1～1000 の場合、6 秒
- CSQ のメッセージ数が 1001～5000 の場合、8 秒
- CSQ のメッセージ数が 5000 を超える場合、10 秒

watt.server.messaging.csq.publishDelayWhileDraining のデフォルト値は 0 です。

watt.server.messaging.csq.publishDelayWhileDraining は、メッセージングプロバイダが Broker または Universal Messaging である webMethods メッセージングに適用されます。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.messaging.debugTrace
Universal Messaging からまたは Digital Event Services 経由のメッセージを受信する webMethods messaging trigger の冗長ログの追加レベルを有効にします。追加ログは、グローバルに、または個々の webMethods messaging trigger レベルで設定できます。

- すべての webMethods messaging triggerに対して debugTrace ログを有効にするには、watt.server.messaging.debugTrace プロパティを「true」に設定します。デフォルト値は「false」です。

  このパラメータの変更を適用するには、トリガーによって使用されるメッセージング接続エイリアスを無効にしてから再度有効にする必要があります。

- 特定の webMethods messaging triggerに対して debugTrace ログを有効にするには、watt.server.messaging.debugTrace プロパティに完全修飾 webMethods messaging trigger 名を追加し、そのプロパティを「true」に設定します。たとえば、「myFolder:myMessagingTrigger」という名前の webMethods messaging triggerに対して debugTrace ログを有効にするには、[**設定**] > [拡張] ページで次のパラメータを追加します。

  watt.server.messaging.debugTrace.myFolder:myMessagingTrigger=true

  このパラメータまたはこのパラメータの変更を適用するには、プロパティで指定した webMethods messaging triggerを無効にしてから再度有効にする必要があります。Integration Server Administrator を使用して個々のトリガーを無効および有効にする方法の詳細については、以下を参照してください。 897 ページの「特定の webMethods Messaging Triggerに対するドキュメント処理の一時停止と再開 」

注:
追加したログをサーバログに記録するには、サーバ機能 0153 ディスパッチャ (Universal Messaging) のログレベルを Trace に設定する必要があります。

watt.server.messaging.deliverNotificationErrors

Integration Server が pub.publish.notifcation.error メッセージを生成して配信するかどうかを指定します。「true」に設定すると、webMethods messaging trigger 処理中にエラーや例外が発生したときに、Integration Server は pub.publish.notification:error メッセージを生成し、配信します。エラーメッセージの生成と配信の結果、Adapter:Error イベントが生成されることがあります。「false」に設定すると、Integration Server は pub.publish.notification:error メッセージを生成または配信しません。この通知エラーメッセージは抑止されます。このプロパティのデフォルト値は「true」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.messaging.trigger.startup.useServerThread
Universal Messaging からメッセージを受信する webMethods messaging trigger を開始、実行するために使用されるスレッドが、ドキュメント処理の [**最大スレッド**] の値により設定されるスレッドの使用制限数の加算対象になるかどうかを指定します。「true」に設定すると、トリガーを実行するスレッドは、ドキュメント処理の [**最大スレッド**] の値により設定されるスレッドの使用制限数に対して加算されません。「false」に設定すると、トリガーを実行してメッセージをリッスンするスレッドは、ドキュメント処理の [**最大スレッド**] の値により設定されるスレッドの使用制限数に対して加算されます。デフォルトは「true」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.metadata.registry.timeout
Integration Server が閉じる前に、CentraSite レジストリへの接続が非アクティブ状態を維持できる分数。Integration Server は、1 つ以上の CentraSite レジストリに接続して、Integration Server アセットのメタデータのパブリッシュや取り消しを行います。一定期間 CentraSite レジストリでアセットのパブリッシュや取り消しが行われない場合、Integration Server はそのレジストリへの接続を閉じます。デフォルトの期間は 10 分です。

watt.server.metering.simulator.enabled
メータリングシミュレータを有効にするかどうかを指定します。このパラメータを「true」に設定すると、Integration Server はトップレベルサービスのメータリングデータを収集し、メータリングシミュレーションメッセージをサーバログに書き込みます。「false」に設定すると、メータリングシミュレータは無効化されます。デフォルトは「false」です。

メータリングシミュレータの詳細については、を参照してください。

watt.server.meteringAgent.logInterval
webMethods Metering Server にデータを送信する「リハーサル」を行うシステムタスク Metering Agent Diagnostic Log を実行する間隔 (分単位) を指定します。正の整数を指定します。デフォルトは 60 分です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.mime.decodeHeaders

pub.mime:createMimeData サービスがヘッダーデコーディングを処理する方法について、グローバルなデフォルト動作を指定します。このプロパティには以下のいずれかの値を指定できます。

指定内容	動作
NONE	デフォルトで pub.mime:createMimeData サービスが MIME ヘッダーまたは本文パートのヘッダーをデコードしないことを示します。これがデフォルトです。
ONLY_MIME_HEADER	デフォルトで pub.mime:createMimeData サービスが MIME ヘッダーのみをデコードすることを示します。
ONLY_BODY_PART_HEADERS	デフォルトで pub.mime:createMimeData サービスが本文パートのヘッダーのみをデコードすることを示します。
BOTH	デフォルトで pub.mime:createMimeData サービスが MIME ヘッダーおよび本文パートのヘッダーをデコードすることを示します。

注:
値の大文字と小文字は区別されません。

注:
pub.mime:createMimeData サービスで *decodeHeaders* 入力パラメータに値が指定されていない場合、Integration Server は watt.server.mime.decodeHeaders の値を使用して、MIME メッセージのヘッダーのみをデコードするかどうかを決定します。pub.mime:createMimeData サービスおよび *decodeHeaders* 入力パラメータの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.mime.largeDataThreshold
pub.mime サービスが一時スペース (Tspace) を使用するタイミングを決めるためのバイト数を指定します。データがこの値よりも大きい場合、pub.mime:addBodyPart はデータを Tspace に保存します。さらに、pub.mime:getEnvelopeStream サービスが *envStream* の代わりに *mimeMessage* を返します。デフォルト値は 26214400 (25 MB) です。

watt.server.mqtt.producer.maxInflight
同じ MQTT 接続エイリアスを使用して一度にパブリッシュできる MQTT メッセージの最大数を指定します。デフォルトは -1 です。この場合、Integration Server は Paho クライアントを使用して、最大同時実行数プロパティを設定します。Paho は最大同時実行数プロパティのデフォルト値として 10 を使用します。設定パラメータによって設定された値を最大同時実行数プロパティの値として使用するには、watt.server.mqtt.producer.maxInflight プロパティを 0 より大きい値に設定します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.netEncoding

サーバがテキストをネットワークに読み書きするために使用するエンコーディングを指定します。この設定は、特定のエンコーディングで明示的にエンコードされたテキストには影響を与えません。デフォルトは「UTF-8」です。pub.client:soapHTTP サービスの *encoding* パラメータを設定した場合、その値が watt.server.netEncoding の設定よりも優先されます。pub.client:soapHTTP の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

pub.client:http サービスが認証のパスワードダイジェストをサブミットし (つまり、*auth/type* フィールドが `Digest` に設定されており)、HTTP サーバ応答にヘッダーフィールド「Content-Typ」が含まれているにもかかわらず文字セットパラメータが含まれていない場合、Integration Server はデフォルト文字セットとして watt.server.netEncoding サーバ設定パラメータを使用します。

watt.server.new.http.session.context
pub.client:http サービスの実行時に Integration Server で新しいセッションオブジェクトを作成するかどうかを指定します。このプロパティを「`true`」に設定した場合、Integration Server は pub.client:http の前回呼び出し時のセッションオブジェクトの値を無視し、新しいセッションオブジェクトを作成します。このプロパティを「`false`」に設定した場合、Integration Server は既存のセッションオブジェクトの値を使用します。watt.server.new.http.session.context 値は pub.client:http サービスの *newSession* パラメータによって上書きできます。デフォルト値は「`false`」です。

watt.server.noObjectURL
ユーザが要求するドキュメントをサーバが見つけることができないために Integration Server Administrator へのログオン実行が 3 回失敗した場合に、サーバが要求をリダイレクトする URL を指定します。デフォルトでは、「**このようなオブジェクトは存在しません**」というメッセージが HTML ページ上に表示されます。

watt.server.noAccessURL
要求されたドキュメントへのアクセス権限をユーザが持たないために Integration Server Administrator へのログオン実行が 3 回失敗した場合に、サーバが要求をリダイレクトする URL を指定します。デフォルトでは、「**アクセスが拒否されました**」というメッセージが HTML ページ上に表示されます。

watt.server.ns.backupNodes
サービスを削除するときに、完全に削除するかどうかを指定します。「`true`」に設定すると、サービスを削除するときに、サービス node.ndf ファイルの名前を node.bak に変更します。デフォルトは「`false`」です。

watt.server.ns.dependencyManager
依存性検査機能を有効にするか、無効にするかを指定します。「`true`」に設定すると、他のエレメントの移動、名前の変更、削除によって影響を受けるエレメントが識別されます。実稼動環境における最適化を図るため、このパラメータを「`false`」に設定することもできます。このパラメータのデフォルトは「`true`」です。

watt.server.ns.decodeJavaService
Integration Server が非 ASCII Unicode データをデコードし、Designer が Java サービスの本文に表示するかどうかを指定します。Java サービスを保存するとき、Integration Server は非 ASCII Unicode 文字を ASCII Unicode エスケープシーケンスとしてエンコードします。デフォルトで、Integration Server はエスケープシーケンスをデコードしません。したがって、Integration Server が Java サービスコードを Designer に送信した場合、元の Unicode 文字で

はなく、エスケープシーケンスが Designer に表示されます。Integration Server でエスケープシーケンスをデコードし、エスケープシーケンスではなく元の Unicode 文字を Designer に表示する場合、「true」に設定します。デフォルトは「false」です。

watt.server.ns.hideWmRoot
Integration Server が WmRoot パッケージを Software AG Designer ワークスペースで非表示にするかどうかを指定します。「`true`」 (デフォルト) に設定すると、Designer は WmRoot パッケージをワークスペースで表示しません。

watt.server.ns.lockingMode
Integration Server でファイルロックを有効化するかどうかを指定します。

- Integration Server でローカルロックを有効化するには、この値を「`full`」に設定します。これがデフォルト値です。
- ユーザロックを無効化し、ロックを表示しない場合は、この値を「`none`」に設定します。
- ユーザロックを無効化するが、システムロックを表示する場合は、この値を「`system`」に設定します。この値は、Local Service Development 機能を使用するために必要です。

watt.server.ns.logDuplicateDocTypeRegistrationAsError
ドキュメントタイプの重複するユニバーサル名の登録に関連するエラーメッセージのログを抑止するか継続するかを示します。

同じ WSDL ドキュメントを使用して、複数のコンシューマ Web サービス記述子および複数の最初の WSDL プロバイダ Web サービス記述子を生成する場合、Integration Server は同じ明示的ユニバーサル名を持つ複数のドキュメントタイプを作成します。Integration Server が重複するユニバーサル名登録を複数回指定してエラーメッセージをログに記録するのは (そのため、ログファイルが圧迫されることがあります)、これらのドキュメントタイプを含むパッケージをロードするときです。

パラメータの値を「false」に指定すると、エラーメッセージのログは抑止されます。「true」に設定すると、ログが再開します。このパラメータのデフォルト値は「true」です。

> **注:**
> watt.server.ns.logErrorsOnRegisteringMultipleDocTypesForAUniversalName を「false」に設定した場合、重複するユニバーサル名に関するエラーメッセージが抑止されるだけです。重複する名前は解決されません。

watt.server.oauth.approvalpage.footer
Integration Server の [OAuth] 承認ページに表示されるフッター情報を指定します。デフォルトはありません。

watt.server.oauth.approvalpage.header
Integration Server の [OAuth] 承認ページに表示される見出し情報を指定します。デフォルトは「`Resource access approval`」です。

watt.server.oauth.approvalpage.logo
Integration Server の [OAuth] 承認ページに表示されるイメージファイルの URL を指定します。イメージファイルを指定する場合は、以下の点に留意してください。

- イメージファイルの幅に制限はありませんが、高さは 92 ピクセル以下である必要があります。

- ブラウザに表示できる任意のイメージファイル形式のイメージ (.gif、.png、.jpeg、別のイメージファイル形式など) を指定できます。
- イメージファイルが Integration Server 上に存在する必要はありません。イメージファイルが別のサーバでホストされている場合は、絶対パスを指定します。

デフォルトは `/WmPublic/images/fw_logo_sag.gif` であり、これは 234x92 ピクセルです。

watt.server.oauth.approvalpage.title
Integration Server の [OAuth] 承認ページに表示されるタイトルを指定します。デフォルトは「`Access Approval`」です。

watt.server.oauth.authCode.expirySeconds
OAuth 許可コードが期限切れになるまでの時間 (秒) を指定します。値 -1 は許可コードが期限切れにならないことを示します。最大値は 2147483647 です。デフォルトは 600 です。

watt.server.oauth.authCode.expirySeconds パラメータの値は、Integration Server Administrator の [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**認証コード有効期間**] フィールドの値に関連付けられています。OAuth 許可コードの有効期限の設定の詳細については、637 ページの「OAuth の設定」を参照してください。

watt.server.oauth.authServer.alias
Integration Server が認証サーバとして使用するリモートサーバのエイリアスを指定します。この値は、Integration Server Administrator の [**外部サーバ**] > [リモートサーバ] ページの [**エイリアス**] フィールドに一致する必要があります。たとえば、「`local`」と入力します。

> **注:**
> watt.server.oauth.authServer.alias でエイリアスを指定する前に、Integration Server Administrator の **[外部サーバ] > [リモートサーバ]** ページでリモートサーバを追加しておく必要があります。

watt.server.oauth.authServer.alias パラメータの値は、Integration Server Administrator の [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**認証サーバ**] フィールドの値に関連付けられています。Software AG では、watt.server.oauth.authServer.alias プロパティを使用する代わりに、Integration Server Administrator を使用して、認証サーバとして使用するリモートサーバエイリアスを指定することをお勧めします。リモートサーバエイリアスの指定の詳細については、637 ページの「OAuth の設定」を参照してください。

watt.server.oauth.custom.responseHeader
「true」に設定すると、クライアントアプリケーションとエンドユーザに返される OAuth および OpenID エラーが HTTP 応答ヘッダーに X-OAuth-Error ヘッダー属性名と共に表示されます。大部分の OAuth および OpenID エラーでは、Integration Server は 400～599 の範囲のステータスコードを返し、エラーメッセージは HTTP 応答の本文で返します。一部のブラウザでは、ステータスコードが成功 (200～299) でない場合は、応答本文は表示されません。OAuth または OpenID を使用するアプリケーションでの問題は、エラーメッセージが応答ヘッダーに表示されると簡単にトラブルシューティングできます。サーバログ機能「0038 HTTP Header」が TRACE に設定されている場合は、サーバログは応答ヘッダーに表示されます。watt.server.oauth.custom.responseHeader が「true」の場合は、OAuth および OpenID エラーメッセージはサーバログの X-OAuth-Error 応答ヘッダー属性として表示されます。

watt.server.oauth.custom.responseHeader のデフォルト値は「false」です。

watt.server.oauth.disableClient.disableTokens

無効なクライアントアカウントに発行されたクライアントアクセストークンを使用してリソースにアクセスできるかどうかを指定します。「true」に設定した場合、Integration Server イントロスペクションエンドポイント pub.oauth:introspectToken サービスでは、トークン発行先のクライアントアカウントが無効であると、トークンは非アクティブであると見なします。サービスは *active* 出力パラメータに対して「false」の値を返します。OAuth リソースサーバは、Integration Server か別のベンダかに関係なく、要求されたリソースへのアクセスを許可しません。「false」に設定した場合、Integration Server イントロスペクションエンドポイントでは、アクセストークンの評価時に、アクセストークン発行先のクライアントアカウントの有効/無効状態を考慮に入れません。デフォルトは「true」です。

watt.server.oauth.log.authErrors
Integration Server がエラーログに OAuth 認証の失敗を書き込むかどうかを指定します。「true」に設定すると、Integration Server がエラーログに OAuth 認証の失敗を書き込みます。「false」に設定すると、Integration Server はどのログにも OAuth 認証の失敗を書き込みません。デフォルトは「false」です。

watt.server.oauth.requireHTTPS
Integration Server で、クライアントアプリケーションが HTTPS を使用して pub.oauth* サービスにアクセスする必要があるかどうかを指定します。このプロパティの値を「`false`」に設定した場合、Integration Server はクライアントアプリケーションが HTTP を使用して pub.oauth サービスにアクセスすることを許可します。デフォルトは「`true`」です。

このプロパティの値を「`true`」に設定している場合に、クライアントアプリケーションが HTTP 経由で pub.oauth サービスのいずれかにアクセスすると、Integration Server は HTTP 400 エラー応答をクライアントに発行し、サービス例外をエラーログに書き込みます。

注:
OAuth 2.0 Authorization Framework に適合するには、このプロパティを「`true`」に設定する必要があります。

重要:
開発を簡素化するために watt.server.oauth.requireHTTPS を「`false`」に設定できますが、実稼動時には OAuth Framework に従って HTTPS を必要とします。HTTPS を要求しない場合、アクセストークンは認証サーバによってクリアテキストで送信され、盗難に対して脆弱になります。

watt.server.oauth.requireHTTPS パラメータの値は、Integration Server Administrator の [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**HTTP が必要**] フィールドの値に関連付けられています。OAuth サービスにアクセスするための HTTPS の要件の詳細については、637 ページの「OAuth の設定」を参照してください。

pub.oauth サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.oauth.requirePKCE
認証コードの譲渡を使用するすべての公開 OAuth クライアントに対して PKCE が必要かどうかを指定します。「true」に設定した場合、すべての公開 OAuth クライアントは *code_challenge* 値を認証エンドポイント (pub.oauth:authorize サービス) に指定し、*code_verifier* 値をトークンエンドポイント (pub.oauth:getToken サービス) に指定する必要があります。
watt.server.oauth.requirePKCE が「true」に設定されていない場合 (たとえば、値が「false」または null である場合)、公開 OAuth クライアントが追加入力を指定する必要はありません。

ただし、*code_challenge* を認証エンドポイントに送信するすべての公開クライアントは、Integration Server の PKCE 機能を使用して認証コード横取り攻撃を回避します。クライアントからトークンエンドポイントへの後続の要求には、*code_verifier* 入力パラメータを含める必要があります。

watt.server.oauth.requirePKCE のデフォルト値は「false」です。

watt.server.oauth.requirePKCE パラメータの値は、Integration Server Administrator の [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**PKCE を必須にする**] フィールドの値に関連付けられています。PKCE の必須化の詳細については、643 ページの「認証コードの譲渡で PKCE を使用する」を参照してください。

watt.server.oauth.requirePost
Integration Server で、クライアントアプリケーションが HTTP POST メソッドを使用して pub.oauth* サービスにアクセスする必要があるかどうかを指定します。このプロパティの値を「`false`」に設定した場合、Integration Server はクライアントアプリケーションが HTTP GET メソッドを使用して pub.oauth サービスにアクセスすることを許可します。デフォルトは「`true`」です。

このプロパティの値を「`true`」に設定している場合に、クライアントアプリケーションが POST 以外の HTTP メソッドを使用して pub.oauth サービスのいずれかにアクセスすると、Integration Server は HTTP 400 エラー応答をクライアントに発行し、サービス例外をエラーログに書き込みます。

注:
OAuth 2.0 Authorization Framework に適合するには、このプロパティを「`true`」に設定する必要があります。

重要:
開発を簡素化するために watt.server.oauth.requirePOST を「`false`」に設定できますが、実稼動時には OAuth Framework に従って HTTP POST メソッドを必要とします。

watt.server.oauth.token.defaultExpirySeconds
アクセストークンが期限切れになるまでの有効期間 (秒) を指定します。値 -1 は許可トークンが期限切れにならないことを示します。最大値は 2147483647 です。デフォルトは 3600 です。

watt.server.oauth.token.defaultExpirySeconds パラメータの値は、Integration Server Administrator の [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**アクセストークン有効期間**] フィールドの値に関連付けられています。[**アクセストークン有効期間**] フィールドの詳細については、637 ページの「OAuth の設定」を参照してください。

watt.server.oauth.token.endpoint.auth
トークンエンドポイントが認証に対して既存のセッションまたは要求を受け入れるかどうかを指定します。「セッション」に設定すると、トークンエンドポイントサービスが Integration Server にアクティブなセッションがあるクライアントからの要求を受け入れることを示します。これらのクライアントが Cookie 要求ヘッダーに有効なセッションを提示する場合、クライアントは pub.oauth:getToken サービス、pub.oauth:getAccessToken サービス、および pub.oauth:refreshAccessToken サービスを使用するためのクレデンシャルを提供する必要はありません。新しいアクセストークンが要求されるたび、または既存のアクセストークンがリフレッシュされるたびに pub.oauth:getToken サービス、pub.oauth:getAccessToken サービス、および pub.oauth:refreshAccessToken サービスを呼び出して、クライアントが認証要求

ヘッダーのクレデンシャルを提供する必要がある場合、「クレデンシャル」に設定します。デフォルト値は「セッション」です。

watt.server.oauth.token.endpoint.auth パラメータの値は、Integration Server Administrator の [**セキュリティ**] > [OAuth] > [OAuth グローバル設定の編集] ページの [**トークンエンドポイント認証**] フィールドの値に関連付けられています。.[トークンエンドポイント認証] フィールドの詳細については、637 ページの「OAuth の設定」を参照してください。

watt.server.oauth.token.endpoint.internal.requireSecret
OAuth トークンエンドポイントサービスが直接呼び出されたとき (OAuth 認証サーバとして機能している Integration Server と同じサーバ上にあるため)、OAuth トークンエンドポイントサービスを呼び出す機密クライアントによって提供される認証コードが機密クライアントに発行されたことを Integration Server で確認するかどうかを指定します。「true」に設定した場合、Integration Server は要求元クライアントによって提供されたクレデンシャルを使用して、OAuth トークンを要求するクライアントによって提供された認証コードが、認証コードを要求したクライアントのクレデンシャルに一致するかどうかを確認します。client_id と client_secret がトークンエンドポイントサービスに渡される入力パイプラインに存在する必要があります。「false」に設定した場合、Integration Server は、クライアントによって提供された認証コードが、トークンを実際に要求したクライアントのクレデンシャルに一致するかどうかを確認しません。デフォルトは、より安全性が低い「false」です。

注:
初期アクセストークンを取得する場合、およびリフレッシュトークンを使用して新しいアクセストークンを取得する場合、watt.server.oauth.token.endpoint.internal.requireSecret の値が適用されます。

注:
リダイレクトエンドポイントとして機能する Integration Server と同じサーバでトークンエンドポイントサービスを呼び出す公開クライアントは、トークンエンドポイントサービスに渡される入力パイプラインにその client_id を引き続き含める必要があります。

HTTP または HTTPS を使用して OAuth トークンエンドポイントからアクセストークンを要求するクライアントは、それまでと同様に引き続きその要求を送信する必要があります。

- 機密クライアントは HTTP Authorization ヘッダーで固有の client_id と client_secret を使用する必要があります。
- 公開クライアントの client_id は要求の本文に含まれる必要があります。

注:
OAuth トークンエンドポイントとして機能するサービスは Integration Server 10.1 では pub.oauth:getAccessToken、それより後のリリースでは pub.oauth:getToken です。

watt.server.package.maxSizeMB
インストール可能なパッケージの最大展開サイズを MB 単位で指定します。展開サイズがこのパラメータの値より大きいパッケージはインストールできません。デフォルト値は 10000 (10 GB) です。

watt.server.package.parallel.threads
Integration Server 起動時の IS パッケージのロードで使用する最大スレッド数を指定します。スレッドは、Integration Server 起動時のパッケージのロード専用として使用されるスレッドプールに含まれます。Integration Server でパッケージを順次にロードする場合、値を 1 に設

定します。Integration Server でパッケージを並行してロードする場合、値を 2 以上に設定します。10 を超えないようにすることをお勧めします。デフォルト値は 6 です。

> **注:**
> watt.server.package.parallel.threads を 0 または負の数に設定した場合、Integration Server はパッケージを順次にロードします。

watt.server.package.pre82WSD.loadExternalResources
パッケージのロード時に、Integration Server がコンシューマ Web サービス記述子または [**8.2 より前のバージョンとの互換モード**] が「true」に設定されている Integration Server のバージョン 8.2 以前で作成された最初の WSDL プロバイダ Web 記述子用に外部リソースをロードするかを指定します。

このパラメータを「true」に設定すると、Integration Server は、パッケージロード時に、コンシューマ Web サービス記述子または最初の WSDL プロバイダ Web サービス記述子用にソース WSDL ドキュメントと関連する XML スキーマ定義ファイル内のすべての import および include ステートメントを解決します。

このパラメータを「false」に設定すると、Integration Server は Web サービス記述子の作成に使用されるソース WSDL ドキュメントに含まれる外部リソースのロードをスキップします。一部の環境では、このパラメータを「false」に設定すると、Web サービス記述子のロードの速度が上がります。デフォルトは「true」です。

> **注:**
> Integration Server 7.1 で導入された Web サービス実装が Integration Server 10.4 以降は廃止されたため、このパラメータは廃止されました。Integration Server バージョン 7.1 で導入された Web サービス実装は 8.2 より前の互換モードで実行される Web サービス記述子を処理します。

watt.server.partner
ハブサーバの IP アドレスまたはドメイン名を指定します。この設定を使用すると、パートナのサーバがリモートの Integration Server に対してリモート呼び出し要求を発行できるようになります。ポート番号を IP アドレスの一部として指定した場合、Integration Server はそのポート番号を無視します。

watt.server.password.historyLength
Integration Server がユーザの履歴に保存する以前設定されたパスワードの最大数を指定します。ユーザアカウントのメモリに保存されているパスワードと一致するパスワードを選択することはできません。デフォルト値は 0 です。

watt.server.password.historyLength は、Integration Server Administrator の [セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集] ページの [記録する古いパスワードの数] フィールドに指定された値を表示します。

watt.server.password.maxIdenticalCharsInARow
パスワードに入れることができる同一文字の連続使用の最大値を指定します。ユーザがパスワードを作成または変更するときは、作成するパスワードが、このパラメータで指定される同一文字の連続使用の上限を超えないようにする必要があります。新しいパスワードで同一文字の連続使用数が上限を超える場合、パスワード施行モードに基づき、Integration Server は watt.server.password.mode サーバ設定パラメータに指定された値に基づいてユーザに警告またはエラーメッセージを送信します。デフォルトは 3 です。

watt.server.password.maxIdenticalCharsInARow は、Integration Server Administrator の [セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集] ページの [連続する同一文字数の最大値] フィールドに指定された値を示します。

watt.server.password.maxLength
パスワードの最大文字数 (アルファベット文字、数字および特殊文字の組み合わせ) を指定します。ユーザがパスワードを変更するときは、作成するパスワードの文字の最大数がこのパラメータで指定される値を超えないようにする必要があります。値が超えた場合、Integration Server は watt.server.password.mode サーバ設定パラメータに指定された値に基づいてユーザに警告またはエラーメッセージを送信します。デフォルトは 64 文字です。

watt.server.password.maxLength は、Integration Server Administrator の **[セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集]** ページの [**パスワードの長さの最大値**] フィールドに指定された値を示します。

watt.server.password.minDigits
Administrator 以外のユーザについて、パスワードに最低限入れる必要がある数字の数を指定します。Administrator 以外のユーザがパスワードを変更するときは、作成するパスワードに、このパラメータで指定される数以上の数字が含まれる必要があります。数字の数の最小値に達しない場合、Integration Server は watt.server.password.mode サーバ設定パラメータに指定された値に基づいてユーザに警告またはエラーメッセージを送信します。デフォルトは 1 です。

watt.server.password.minDigits は、Integration Server Administrator の **[セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集]** ページの [**数字の数の最小値**] フィールドに指定された値を示します。

watt.server.password.minLength
Administrator 以外のユーザについて、パスワードに最低限入れる必要がある文字の数を指定します。文字の長さには、アルファベットの大文字、小文字、数字 (0～9)、および特殊文字の組み合わせが含まれます。Administrator 以外のユーザがパスワードを変更するときは、作成するパスワードに、このパラメータで指定される数以上の文字が含まれる必要があります。最小限の長さに達しない場合、Integration Server は watt.server.password.mode サーバ設定パラメータに指定された値に基づいてユーザに警告またはエラーメッセージを送信します。デフォルトは 8 です。

watt.server.password.minLength は、Integration Server Administrator の [セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集] ページの [パスワードの長さの最小値] フィールドに指定された値を示します。

watt.server.password.minLowerChars
Administrator 以外のユーザについて、パスワードに最低限入れる必要があるアルファベットの小文字の数を指定します。Administrator 以外のユーザがパスワードを変更するときは、作成するパスワードに、このパラメータで指定される数以上の小文字が含まれる必要があります。最小数に達しない場合、Integration Server は watt.server.password.mode サーバ設定パラメータに指定された値に基づいてユーザに警告またはエラーメッセージを送信します。デフォルトは 2 です。

watt.server.password.minLowerChars は、Integration Server Administrator の [セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集] ページの [アルファベットの小文字数の最小値] フィールドに指定された値を示します。

watt.server.password.minSpecialChars

Administrator 以外のユーザについて、パスワードに最低限入れる必要がある特殊文字 (*、.、?など) の数を指定します。Administrator 以外のユーザがパスワードを変更するときは、作成するパスワードに、このパラメータで指定される数以上の特殊文字が含まれる必要があります。最小値が満たされない場合、Integration Server は Administrator ユーザに警告メッセージを送信します。デフォルトは 2 です。

注:
パスワードの先頭文字にアスタリスク (*) を使用することはできません。

watt.server.password.minSpecialChars は、Integration Server Administrator の [セキュリティ] > [ユーザ管理] > [パスワード制限] > [パスワード制限の編集] ページの [特殊文字数の最小値] フィールドに指定された値を示します。

watt.server.password.minUpperChars
Administrator 以外のユーザについて、パスワードに最低限入れる必要があるアルファベットの大文字の数を指定します。Administrator 以外のユーザがパスワードを変更するときは、作成するパスワードに、このパラメータで指定される数以上の大文字が含まれる必要があります。最小数に達しない場合、Integration Server は watt.server.password.mode サーバ設定パラメータに指定された値に基づいてユーザに警告またはエラーメッセージを送信します。

watt.server.password.mode
Administrator および Administrator 以外のユーザによってパスワードが設定される際に、パスワード制限を設定するかしないかを指定します。「`strict`」に設定するとパスワード制限がすべてのユーザに実行され、「`lax`」に設定すると Administrator 以外のユーザにパスワード制限が実行されます。デフォルトは「`lax`」です。

watt.server.pipeline.processor
[パイプラインデバッグ] 機能をグローバルに有効にするか、無効にするかを指定します。「`true`」に設定した場合、Integration Server は実行時にサービスの [パイプラインデバッグ] 値をチェックします。[パイプラインデバッグ] 値は、Designer の [プロパティ] ビューで表示および設定できます。「`false`」に設定した場合、Integration Server は Designer でサービスに設定されている [パイプラインデバッグ] オプションを無視します。デフォルトは「`true`」です。

テストおよびデバッグが実行される開発環境では、このプロパティを有効にします。ただし、実稼動環境では、このプロパティを無効にすることをお勧めします。

重要:
新しい値を適用するには、Integration Server を再起動する必要があります。

watt.server.ping
pub.remote:invoke によって作成された Pinger スレッドが wm.sever.ping 要求を送信する間隔をミリ秒単位で指定します。watt.server.ping.interval の値は、リモートサーバエイリアスの [**キープアライブタイムアウト**] フィールドが指定されていないときのみ使用されます。デフォルトは 1000 ミリ秒です。

watt.server.port
Integration Server のプライマリポートのポート番号を指定します。デフォルトは `5555` です。

watt.server.portAccess.axis2
Axis2 ベースの Web サービスにポート経由でアクセスできることを Integration Server で確認するかどうかを指定します。Integration Server で Axis2 ベースの Web サービスのアクセ

スチェックを行う場合、このパラメータを「true」に設定します。Axis2 ベースの Web サービスのアクセスチェックを無効にするには、このパラメータを「false」に設定します。デフォルトは「true」です。

watt.server.portQueue
HTTP および HTTPS ポートのポートキューのサイズを指定します。ポートキューは、TCP/IP スタックにキューイングされている未解決の受信接続の数を表します。デフォルトは 65534 です。サーバが AS/400 上で動作している場合は、この番号を 511 に設定します。

watt.server.ports.ipaccess.ignoreXForwardedForHeader
Integration Server が、ポートへのアクセスを許可するか、拒否するかを決定するときに、要求の X-Forwarded-For (XFF) ヘッダーの IP アドレスを使用するか、無視するかを指定します。このパラメータは、Enterprise Gateway 外部ポートと API Gateway 外部ポートで受信された要求のみに適用されます。

「true」に指定されている場合は、要求がポートにアクセスを許可されたホストからかどうかを判断するために、Integration Server は XFF ヘッダー内の IP アドレスを使用しません。代わりに、Integration Server はプロキシまたは負荷分散機能の IP アドレスを使用します。

- 要求が、デフォルトで許可に設定されているポートの拒否リストに IP アドレスが含まれているホストからプロキシまたは負荷分散機能を介して送信され、プロキシまたは負荷分散機能の IP アドレスがブロックされていない場合、このホストはブロックされません。つまり、このホストはポートにアクセスすることになります。
- 要求が、デフォルトで拒否に設定されているポートの許可リストに IP アドレスが含まれているホストからプロキシまたは負荷分散機能を介して送信され、プロキシまたは負荷分散機能の IP アドレスがブロックされている場合、このホストはポートにアクセスできません。

「false」に指定されている場合は、要求がポートにアクセスを許可されたホストからかどうかを判断するために、Integration Server は XFF ヘッダー内の IP アドレスを使用します。

- 要求が、デフォルトで許可に設定されているポートの拒否リストに IP アドレスが含まれているホストからプロキシまたは負荷分散機能を介して送信された場合、このホストはポートにアクセスできません。
- 要求が、デフォルトで拒否に設定されているポートの許可リストに IP アドレスが含まれているホストからプロキシまたは負荷分散機能を介して送信された場合、このホストはポートにアクセスできます。

デフォルト値は「true」です。

watt.server.portStateless
ステートレスである Integration Server のポートのポート番号のカンマ区切りリストを指定します。Integration Server は、このポートで受信された要求のセッションまたはセッション ID を提供および保持しません。

プライマリポートおよび診断ポートを除く、Integration Server の任意の HTTP ポートまたは HTTPS ポートをステートレスに設定できます。また、WebSockets 用、またはIntegration Server の管理に使用するポートはステートレスに設定しないでください。プライマリポート、診断ポート、HTTP ポートまたは HTTPS ポートではないポート、または watt.server.portStateless の値に存在しないポートを指定した場合、Integration Server では、ポートをステートレスにできずスキップすることを示す警告メッセージをログに記録します。

このパラメータのデフォルト値はありません。つまり、デフォルトでは、Integration Server にステートレスのポートはありません。

watt.server.pubDateTimeFormat.javaSlidingWindow
pub.date:dateTimeFormat サービスで年に 2 デジットパターンである *yy* を使う場合に使用するスライディングウィンドウを指定します。「true」に設定されている場合、pub.date:dateTimeFormat サービスは、Java クラス java.text.SimpleDateFormat に記述されているように、現在の年に対して前 80 年、後 20 年を採用します。「false」に設定されている場合、pub.date:dateTimeFormat サービスは現在の年に対して前後 50 年づつのスライディングウィンドウを使用します。デフォルトは「false」です。

watt.server.publish.drainCSQInOrder
クライアントサイドキューを有効にしている場合に、Integration Server で送信ドキュメントストアをパブリケーション順序で空にするか、新規にパブリッシュされたドキュメントと並行して空にするかを指定します。このパラメータを「`true`」に設定した場合、Integration Server は、送信ドキュメントストアが空になるまで、新規にパブリッシュされたすべてのドキュメント (保証付きドキュメントおよび揮発性ドキュメント) を送信ドキュメントストアに送信します。この処理を行うことで、Integration Server はパブリケーション順序を維持できます。このパラメータを「`false`」に設定した場合、Broker への新しいドキュメントのパブリッシュ中に送信ストアが空になります。デフォルトは「`true`」です。

注:
このサーバ設定パラメータは、Broker にパブリッシュされるドキュメントのみに適用されます。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.publish.local.rejectOOS
サブスクライブトリガーのキューが最大容量に達した場合に、ローカルにパブリッシュされたドキュメントを Integration Server が拒否するかどうかを指定します。

このパラメータを「`true`」に設定すると、Integration Server は、ローカルにパブリッシュされたドキュメントをサブスクライブトリガーのキューに入れる前に、トリガーのキューサイズをチェックします。トリガーの [**容量**] プロパティで指定された最大数のドキュメントが既にキューに入っている場合、Integration Server は、このトリガーキューについてのみ、ローカルにパブリッシュされたドキュメントを拒否します。

このパラメータを「`false`」に設定した場合、Integration Server は、キューが最大容量に達した後も、ローカルにパブリッシュされたドキュメントをサブスクライブトリガーのキューに入れ続けます。これがデフォルトです。

注:
複数のトリガーが同じドキュメントをサブスクライブできます。Integration Server は、最大容量に達していない任意のサブスクライブトリガーキューにドキュメントを配置します。

注:
このパラメータは、pub.publish:publish または pub.publish.publishAndWait サービスを使用してローカルにパブリッシュされたドキュメントにのみ適用されます。

watt.server.publish.maxCSQRedeliveryCount

クライアントサイドキュー (CSQ) から Broker にメッセージをパブリッシュするときに Integration Server が実行する最大再デリバー試行回数を指定します。Integration Server がメッセージをパブリッシュし、Broker が使用可能でないときに、Broker (watt.server.publish.useCSQ = always) と共に使用するよう設定されている場合、Integration Server は保証付きメッセージを CSQ に書き込みます。Broker が使用可能になると、Integration Server は、CSQ から Broker へメッセージをパブリッシュすることによって CSQ を排出します。CSQ から Broker へメッセージをパブリッシュする最初の試行が失敗した場合、Integration Server はメッセージが正常にパブリッシュされるまで、または watt.server.publish.maxCSQRedeliveryCoun で指定された最大回数まで再試行を実行します。CSQ から Broker にパブリッシュする各試行は再デリバー試行と見なされます。watt.server.publish.maxCSQRedeliveryCount パラメータには正の整数を設定する必要があります。デフォルト値は 5 です。

注:
このパラメータは、Broker がメッセージングプロバイダである webMethods メッセージングに適用されます。Universal Messaging がメッセージングプロバイダである webMethods メッセージングの比較可能なプロパティは watt.server.messaging.csq.maxRedeliveryCount です。プロバイダに関係なく、比較可能なパラメータは watt.server.jms.csq.maxRedeliveryCount です。

注:
Broker が廃止されたため、このパラメータは廃止されました。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.publish.useCSQ
Integration Server を使用できない状況でドキュメントがパブリッシュされた場合に、Broker が送信クライアントサイドキューを使用するかどうかを指定します。Broker を使用できない状況でパブリッシュされたドキュメントを送信ドキュメントストアに送信する場合は、このパラメータを「`always`」に設定します。Broker が使用不可の場合は ServiceException をスローするようにパブリッシャーサービスに通知する場合は、このパラメータを「`never`」に設定します。デフォルトは「`always`」です。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.publish.usePipelineBrokerEvent
Integration Server がドキュメントを Broker に対してパブリッシュする際に通常行われるエンコーディングを省略するかどうかを指定します。このプロパティを「`true`」に設定した場合、Integration Server は、$brokerEvent という名前のオブジェクトをパイプラインから検索します。オブジェクトが検出され、そのタイプが BrokerEvent である場合、Integration Server はこの値を Broker に送信し、これ以上のエンコーディングは実行しません。Integration Server が Broker のネイティブイベントを送信する場合は、このパラメータを「`true`」に設定します。デフォルトは「`false`」です。

Broker ネイティブイベントのパブリッシュの詳細については、『*Publish-Subscribe Developer's Guide*』を参照してください。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.publish.validateOnIS
Integration Server がパブリッシュ済みドキュメントの妥当性検査を行うかどうかを指定します。このパラメータは、次のいずれかの値に設定します。

指定内容	[宛先]
`always`	すべてのパブリッシュ済みドキュメントについて、ドキュメントの妥当性検査を実行します。これには、[**パブリッシュ時の妥当性検査**] プロパティが「`false`」に設定されている、パブリッシュ可能なドキュメントタイプのインスタンスが含まれます。
`never`	パブリッシュ可能なすべてのドキュメントタイプについて、ドキュメントの妥当性検査を無効化します。これには、[**パブリッシュ時の妥当性検査**] プロパティが「`true`」に設定されている、パブリッシュ可能なドキュメントタイプのインスタンスが含まれます。 ドキュメントの妥当性検査の無効化が必要になる理由としては、次のようなものが考えられます。 ■ パフォーマンスを改善する必要がある。 ■ ドキュメントを手動で確認したい。 ■ Integration Server にデータを送信したシステムによってデータの妥当性検査が終了している。 ■ Integration Server ではなく、メッセージングプロバイダを使用してドキュメントの妥当性検査を行いたい。 ■ Integration Server は、ネイティブ Broker イベントを送受信しています。
`perDoc`	ドキュメントの妥当性検査が有効化されている、パブリッシュ可能なドキュメントタイプのインスタンスについてのみ、ドキュメントの妥当性検査を実行します。言い換えるならば、Integration Server は、[**パブリッシュ時の妥当性検査**] プロパティが「`true`」に設定されている、パブリッシュ可能なドキュメントタイプのインスタンスであるパブリッシュ済みドキュメントの妥当性検査を実行します。 これがデフォルトの動作です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

Broker のネイティブイベントの処理、およびパブリッシュ可能な個々のドキュメントタイプの妥当性検査の指定の詳細については、『*Publish-Subscribe Developer's Guide*』を参照してください。

watt.server.recordTodocument.bufferSize

Integration Server でドキュメント (IData) を XML ストリングに変換するときに、XML ストリングを書き込むために使用されるバッファのサイズを指定します。作成される XML ストリングの大半が 4 KB を超える場合、バッファサイズを大きくします。作成される XML ストリングの大半が 4 KB 未満の小さなドキュメントになる場合、バッファサイズを小さくします。デフォルトのサイズは 4 KB です。

watt.server.remoteInvoke.queryCSRFToken
リモートサービス呼び出し中に、Integration Server がリモートサーバで現在のセッションの CSRF トークンをクエリーし、サービス要求にそのトークンを含めるかどうかを示します。Integration Server がクロスサイトリクエストフォージェリ (CSRF) ガードを使用する場合、サーバに送信される要求には CSRF トークンを含める必要があります。要求に CSRF トークンが含まれていない場合、サーバは要求を拒否します。Integration Server がリモート呼び出しを行い、CSRF ガードを使用する別の Integration Server でサービスを実行する場合、要求には CSRF トークンを含める必要があります。要求に CSRF トークンが含まれていることを確認するため、要求元 Integration Server はリモート Integration Server から現在のセッションの CSRF トークンを取得します。次に、要求元 Integration Server は要求を変更して CSRF トークンを含めます。ただし、これは、リモート Integration Server が CSRF ガードを使用している場合にのみ必要です。リモート Integration Server が CSRF ガードを使用し、要求元 Integration Server が現在の CSRF トークンをクエリーし、そのトークンを要求に含めるようにする場合、watt.server.remoteInvoke.queryCSRFToken を「true」に設定します。リモート Integration Server が CSRF ガードを使用しない場合、「false」に設定します。デフォルトは「true」です。

watt.server.removeDefaultMWSRolesInISACL
Integration Server が ACL からデフォルトの MWS 役割 (**My webMethods Administrators** および **My webMethods Users**) を削除するかどうかを指定します。プロパティを「`true`」に設定した場合、Integration Server は ACL からデフォルト MWS 役割を削除します。プロパティを「`false`」に設定した場合、Integration Server は ACL のデフォルト MWS 役割を保持します。デフォルト値は「`false`」です。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動してください。

watt.server.requestCerts
Integration Server が、SSL によって接続されたクライアントからのデジタル認証を要求するかどうかを指定します。サーバで認証を要求する場合は、このパラメータを「`true`」に設定します。サーバで認証を要求しない場合は、「`false`」に設定します。デフォルトは「`false`」です。

watt.server.requireV3Manifest
Integration Server が、非アクティブのパッケージリストを manifest.v3 ファイルを持つパッケージのみに制限するかどうかを指定します。「true」に設定されている場合、Integration Server は、manifest.v3 ファイルを持つパッケージを使用して非アクティブパッケージのリストを構築します。manifest.v3 ファイルを持つパッケージが Integration Server に知られていない場合や、休止していない場合、Integration Server はパッケージを非アクティブであるとみなします。「false」に設定されている場合、Integration Server は、パッケージが非アクティブかどうかを判断するために manifest.v3 ファイルの有無を使用しません。代わりに、Integration Server は、 *Integration Server_directory* /instances/*instanceName*/packages ディレクトリ内のすべてのパッケージに対して、既知であるかどうか、または休止であるかどう

かを確認することにより、非アクティブであるかどうかを確認します。デフォルト値は「true」です。

watt.server.response.displayISErrorCode
エラー状況の場合の Integration Server からクライアントアプリケーションへの HTTP 応答で、応答のヘッダーと本文に Integration Server エラーコードが含まれるかどうかを示します。
このパラメータの値を「`false`」に設定すると、エラー状況時の HTTP 応答のヘッダーと本文の両方に Integration Server エラーコードなしのエラーメッセージテキストが含まれます。値を「`true`」に設定すると、エラー状況時の HTTP 応答のヘッダーと本文に Integration Server エラーコードおよび対応するエラーメッセージテキストが含まれます。このパラメータのデフォルト値は「`true`」です。

watt.server.rest.addHTTPMethodToInputPipeline
Integration Server が、REST 要求を処理中に、要求された HTTP メソッドの入力パイプラインに *$httpMethod* 変数を追加するかどうかを示します。「`true`」に設定されている場合、Integration Server は、*$httpMethod* 入力変数を入力パイプラインに追加します。「`false`」に設定されている場合、Integration Server は、*$httpMethod* 入力変数を入力パイプラインに追加しません。デフォルト値は「`false`」です。

watt.server.rest.removeInputVariablesFromResponse
Integration Server が、REST API 応答の送信中に、パイプラインからサービス署名の入力変数を削除するかどうかを示します。このプロパティが「`true`」に設定されている場合、Integration Server は、Swagger ドキュメントに定義されている応答構造に一致する REST API 応答の出力変数のみを保持します。このプロパティが「`false`」に設定されている場合、Integration Server は、REST API 応答の送信中に入力変数も保持します。デフォルトは「`true`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.RESTDirective
Integration Server のリソースにアクセスする URL の rest ディレクティブに使用する代替語を指定します。デフォルトでは、このパラメータは `watt.server.RESTDirective=rest` と設定され、ユーザは rest ディレクティブを rest (たとえば、`GET http://`*`host:port`*`/rest/`*`resource`*`/resourceID`) と指定する必要があります。rest ディレクティブを `rest` または代替語で指定することをユーザに許可する場合は、このパラメータに代替語を設定します。たとえば、rest ディレクティブを `rest` または `process` のいずれかで指定することをユーザに許可する場合は (`GET http://`*`host:port`*`/rest/`*`resource`*`/resourceID` または `GET http://host:port/process/resource/resourceID`)、このパラメータを `watt.server.RESTDirective=process` と設定します。

重要:

- このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。
- このパラメータの設定を変更した後に Integration Server を再起動すると、パラメータの以前の設定を含み、Integration Server Administrator を使用して作成された URL パスを指す URL エイリアスはすべて***動作しなくなります***。したがって、Integration Server Administrator から影響を受ける各 URL エイリアスの URL パスを編集し、パ

ラメータを更新した設定を含める必要があります。この制限は、Software AG Designer を使用して作成された URL エイリアスには適用されません。

watt.server.RESTDirective.V2
Integration Server のリソースにアクセスする URL の restv2 ディレクティブに使用する代替語を指定します。デフォルトでは、このパラメータは `watt.server.RESTDirective.V2=restv2` と設定され、ユーザは rest ディレクティブを restv2 (たとえば、`GET http://host:port/restv2/resource/resourceID`) と指定する必要があります。rest ディレクティブを `restv2` または代替語で指定することをユーザに許可する場合は、このパラメータに代替語を設定します。たとえば、ディレクティブを `restv2` または `process` のいずれかで指定することをユーザに許可する場合は (`GET http://host:port/restv2/resource/resourceID` または `GET http://host:port/process/resource/resourceID`)、このパラメータを `watt.server.RESTDirective.V2=process` と設定します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.revInvoke.proxyMapUserCerts
webMethods Enterprise Gateway の設定にのみ使用します。Enterprise Gateway Server を使用して、処理のために認証情報を内部サーバに渡すのみでなく、クライアント認証も実行するかどうかを指定します。デフォルトは「`false`」です。このプロパティを「`true`」に設定した場合、要求が内部サーバ上の保護されていないサービスに対するものであっても、Enterprise Gateway Server はすべての匿名要求 (認証もユーザ名/パスワードも提供されていないもの) を拒否します。詳細については、593 ページの「webMethods Enterprise Gateway の設定 」を参照してください。

watt.server.rg.internalregistration.timeout
webMethods Enterprise Gateway 設定にのみ使用します。内部サーバでこの値を設定します。応答がない Enterprise Gateway Server への接続を閉じるまでに内部サーバが待機する時間 (秒) を指定します。デフォルトは 0 です。つまり、内部サーバはタイムアウトしません (無限のタイムアウト期間)。

重要:
内部サーバの watt.server.rg.internalregistration.timeout 値は、Enterprise Gateway Server の watt.net.socketpool.sweeperInterval サーバ設定パラメータで定義した ping 値よりも大きい値に設定してください。watt.server.rg.internalregistration.timeout を watt.net.socketpool.sweeperInterval よりも小さい値に設定すると、内部サーバは Enterprise Gateway Server への接続を閉じて、新しい接続を定期的に再確立します。

watt.server.rg.internalsocket.timeout
Enterprise Gateway Server で、HTTP 500 内部サーバエラーで要求を終了する前に、クライアント要求が内部サーバへの接続まで待機できる時間をミリ秒単位で指定します。指定したタイムアウト期間内に内部サーバへの接続が使用できるようになった場合、Enterprise Gateway Server は要求を内部サーバに転送します。タイムアウトを経過する前に内部サーバが使用できるようにならない場合、Enterprise Gateway Server は HTTP 500 内部サーバエラーを要求元クライアントに返し、エラーログに次のメッセージを書き込みます。

"Enterprise Gateway port {port_number} is unable to forward the request to Internal Server because there are no Internal Server connections available."

このパラメータのデフォルト値は 0 です。つまり、クライアント要求は内部サーバへの接続まで無制限に待機します。

watt.server.scheduler.deleteCompletedTasks
期限切れのスケジュール済みタスクを自動的に削除するかどうかを指定します。「true」に設定した場合、Integration Server は期限が切れたタスクをその時点で削除します。Integration Server の次回の再起動後に期限切れのタスクを削除する場合は、このパラメータを「false」に設定します。デフォルトは「true」です。

watt.server.scheduler.ignoreReferenceValidationErrors
スケジュール済みサービスの参照が破損している場合に、Integration Server がスケジュール済みタスクの作成や実行を行うかどうかを指定します。プロパティを「true」に設定すると、スケジュール済みタスクを実行するときに、Integration Server はスケジュール済みサービスの破損した参照を無視します。プロパティを「false」に設定すると、スケジュール済みサービスに 1 つ以上の破損した参照がある場合に、Integration Server はスケジュール済みタスクを実行しません。デフォルトは「false」です。

watt.server.scheduler.logical.hostname
単一のマシンでホストされているクラスタ内の Integration Server インスタンスを識別する一意の論理名を指定します。デフォルトでは、Integration Server はタスクのスケジュール中にホスト名を使用して自己を識別します。ただし、Integration Server のクラスタが単一のマシンでホストされている場合、ホスト名自体では個々の Integration Server インスタンスを一意に識別できません。このような場合、watt.server.scheduler.logical.hostname プロパティを使用して、個々の Integration Server インスタンスを識別する一意の論理名を指定します。

このパラメータのデフォルト値はホスト名です。

watt.server.scheduler.logical.hostname プロパティを設定するときには、次の点に留意してください。

- このプロパティは、単一のマシン上のクラスタ内で複数の Integration Server を実行している場合にのみ設定します。
- このプロパティは、タスクをスケジュールする前に、クラスタ内の各 Integration Server インスタンスで設定します。
- 一意の論理ホスト名を指定する必要があります。
- セミコロン (;) を含む論理ホスト名は指定できません。

このプロパティは、ホスト名または IP アドレスが経時的に変化する仮想サーバに Integration Server がインストールされている場合にも役立ちます。watt.server.scheduler.logical.hostname を使用して、クラスタ内の各サーバを識別する固定値を指定できます。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.scheduler.maxWait
Integration Server がタスクキューをチェックする間隔 (秒単位) の最大値です。サーバは、実行予定のタスクがないか、タスクキューを定期的にチェックします。実行待ちのタスクがない場合、サーバは maxWait に指定された時間の経過後に、再度キューをチェックします。実行待ちのタスクがある場合、サーバはタスクのスケジュールされた実行時間か maxWait に指定された時間の経過後、いずれか早い方のタイミングで再度キューをチェックします。たと

えば、保留中のタスクの実行が 30 秒後に予定され、maxWait が 60 に指定されている場合、サーバは 30 秒後に再度キューをチェックします。デフォルトは 60 です。

注:
このパラメータで指定した値は、スケジュール済みタスクで設定された最小間隔よりも小さくする必要があります。

Integration Server のクラスタを使用し、クラスタのすべてのサーバ上で実行する 1 つのタスクをスケジュールしている場合、各サーバのプロパティ設定が異なっていれば、サーバごとにタスクの開始時間が異なることになります。このため、クラスタ環境で稼動する場合、クラスタのすべてのサーバでこのプロパティを同じ値に設定する必要があります。Integration Server のクラスタ設定の詳細については、『*webMethods Integration Server Clustering Guide*』を参照してください。

watt.server.scheduler.threadThrottle
スケジューラプロセスが使用できる Integration Server スレッドの割合です。デフォルトは 75% です。

watt.server.securePort
Integration Server のデフォルトのセキュアポート (HTTPS) のポート番号を指定します。インストール中に、Integration Server は診断ポートを自動的に 5543 に作成します。しかし、インスール中に異なる値を指定できます。デフォルトは 5543 です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。Integration Server を再起動せずにデフォルトのセキュアポートを変更するには、**[サーバ] > [ポート]** ページでデフォルトのセキュアポートを編集します。ポートの編集の詳細については、228 ページの「ポートの編集」を参照してください。

watt.server.security.createDefaultServicesOnPorts
Integration Server が内部サービスおよび組み込みサービスのデフォルトリストを、アクセスを拒否するよう設定されたポートの許可リストに追加するかどうかを指定します。一部の内部サービスでは実行 ACL が anonymous に設定されており、これが一部の環境でセキュリティの脆弱性となることがあります。内部サービスと組み込みサービスの事前定義済みリストを許可リストに含める場合は、「true」に設定します。内部サービスまたは組み込みサービスを許可リストに追加しない場合は、「false」に設定します。デフォルトは「true」です。

watt.server.securityLog.eg.enableExternalClientIP
Enterprise Gateway により Integration Server のサービスを呼び出す外部クライアントの IP アドレスをセキュリティログに記録するかどうかを指定します。「true」を設定した場合、Integration Server はセキュリティログに外部クライアントの IP アドレスを記録します。「false」を設定した場合、Integration Server はセキュリティログに Enterprise Gateway の IP アドレスを記録します。デフォルト値は「false」です。

watt.server.securityLog.ignoreXForwardedForHeader
Integration Server が X-Forwarded-For 応答ヘッダーを無視し、セキュリティログのクライアント ID としてプロキシサーバの IP アドレスを使用するかどうかを指定します。「false」に設定した場合、Integration Server は X-Forwarded-For 要求ヘッダーから送信元クライアントの IP アドレスを取得し、セキュリティログおよびセキュリティログメッセージでそのクライアントの IP アドレスを使用します。`watt.server.securityLog.ignoreXForwardedForHeader` を「true」に設定した場合、Integration Server は X-Forwarded-For 要求ヘッダーを無視し、プロキシサーバの IP アドレスをクライアント ID として使用します。要求を発行したクライア

ントの IP アドレスをセキュリティログエントリに含めるほうが、要求が anonymous の場合は特に役に立ちます。デフォルトは「false」です。

watt.server.securityLog.logAnonymousRequests
anonymous の認証要求をセキュリティログに書き込むかどうかを制御します。watt.server.securityLog.logAnonymousRequests を「true」に設定すると、セキュリティロガーは anonymous の認証要求をセキュリティログに書き込みます。ログエントリでは、anonymous 要求のユーザ ID は「local/default」になります。デフォルト値は「false」であり、これはセキュリティロガーに anonymous 認証要求を含めないことを示します。

watt.server.serverlogFilesToKeep
現在のサーバログファイルを含め、Integration Server がファイルシステムに保持するサーバログファイルの数を指定します。Integration Server がサーバログファイル数の制限に達した場合、Integration Server はサーバログを循環するごとに、アーカイブ済みの最も古いログファイルデータを削除します。watt.server.log.filesToKeep を 1 に設定した場合、Integration Server は現在の server.log ファイルのみ保持し、以前の server.log ファイルは保持しません。Integration Server が server.log を循環するとき、Integration Server は以前のサーバログのアーカイブファイルを作成しません。watt.server.log.filesToKeep を 0 または 1 未満の値に設定した場合、Integration Server は無制限の数のサーバログファイルを保持します。

watt.server.log.filesToKeep のデフォルト値は -1 です。これは、Integration Server が保持するサーバログファイル数に制限がないことを意味します。

Integration Server が保持するサーバログファイルの最大数を設定する方法の詳細については、235 ページの「サーバログの設定」を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.serverlogMaskBearerToken
Integration Server がサーバログへの書き込み時にベアラートークンの値をマスクするかどうかを指定します。サーバログ機能「0038 HTTP ヘッダー」が Trace に設定されており、Integration Server がベアラートークンを含む要求を受信する場合、Integration Server は承認要求ヘッダーをサーバログに書き込みます。watt.server.serverlogMaskBearerToken が「true」である場合、Integration Server は実際のベアラートークン値の代わりにアスタリスク文字列をログに書き込みます。watt.server.serverlogMaskBearerToken が「false」である場合、ベアラートークンがログに書き込まれます。デフォルトは「true」です。

watt.server.serverlogRotateSize
Integration Server が server.log ファイルをロールオーバーするサイズを指定します。このプロパティは、*N*[KB|MB|GB] に設定します。*N* は、有効な任意の整数です。Integration Server が server.log ファイルを循環する最小サイズは 33 KB です。単位として KB を使用する場合、*N* は 33 以上の値に設定する必要があります。単位を指定しない場合、Integration Server では *N* 値をバイトとして扱います。この場合、*N* は 32768 以上に設定しないと有効になりません。整数と測定単位の間にスペースを入れないでください。

このパラメータのデフォルト値はありません。デフォルトでは、パラメータの値はありません。watt.server.serverlogRotateSize の値が指定されない場合は、Integration Server は server.log ファイルを午前零時にのみ循環します。

無効な値が指定されると、Integration Server はパラメータを -1 にリセットします。値を -1 にすると、パラメータの値を指定しない場合と同じ動作になります。

server.log の詳細については、235 ページの「サーバログの設定」を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.serverlogQueueSize
サーバのログキューに収容可能なエントリの数を制御します。このプロパティは、サーバログに直接エントリを書き込むか、最初にメモリ内のキューに入れてからバックグラウンドスレッドを使用してサーバログに書き込むかを制御する、`watt.server.log.queued` プロパティと関連します。`watt.server.log.queued` プロパティを「`true`」に設定している場合、サーバログエントリが予想どおりにログに書き込まれていないときには、キューサイズを大きくしてみてください。サーバログとサーバログキューの詳細については、239 ページの「サーバログエントリをキューに格納するかどうかの指定」を参照してください。デフォルトのキューサイズは 8192 です。

watt.server.serverclassloadername
これは内部プロパティです。変更しないでください。

watt.server.service.blacklist
カンマ区切りリストまたはファイルを使用して、サービスブラックリストのサービスと、サービスがサービスブラックリストにあるインタフェースのどちらか一方または両方を指定します。ブラックリストに記載されたサービス、またはブラックリストに記載されたインタフェースのサービスを呼び出そうとすると、アクセス拒否エラーになります。WmPublic や WmRoot のサービスなど、Integration Server で配信されるすべてのサービスはブラックリストに記載できます。Integration Server が機能するために必要不可欠なサービスをブラックリストに記載しないでください。このサーバ設定パラメータのデフォルト値はありません。

Integration Server はこのパラメータでのワイルドカードの使用をサポートしてません。

サービスブラックリストの詳細については、545 ページの「ブラックリストへのサービスの追加」を参照してください。

重要:
このパラメータの設定、またはサービスブラックリストとして使用されるファイルの内容を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.service.lazyEval
Integration Server が起動中に Java サービスを初期化するかどうかを制御します。「`true`」(デフォルト) に設定すると、実行中に Java サービスが参照されるまで、Integration Server は Java サービスを初期化しません。このため、サーバが起動時に多くの Java サービスを初期化する場合のパフォーマンスが向上します。

watt.server.service.list.treatEmptyAsNull
フローサービスの実行中に入力値がない場合に Integration Server が Document List、String List、Document Reference List、Object List など、すべてのリストデータタイプに NULL 値を割り当てるかどうかを指定します。「`true`」は、サービスの出力がリストデータタイプに NULL 値を割り当てることを示します。「`false`」は、サービスの出力が NULL 値を割り当てないことを示します。デフォルトは「`true`」です。

watt.server.serviceMail

Integration Server からサービスの失敗に関するメッセージを送信する電子メールアドレスを指定します。デフォルトはありません。

このパラメータで電子メールアドレスを指定すると、Integration Server は単純なスケジュール済みタスクを作成して通知を実行します。このタスクでは、Integration Server が cronjob ベースのスレッドプールから使用するスレッドが必要になります。このスレッドプールの詳細については、watt.server.cron.maxThreads および watt.server.cron.minThreads パラメータを参照してください。

watt.server.serviceMonitor.queryOnServerId
クラスタの 1 つのインスタンスだけでなく、ステートフルクラスタ全体の監視機能を指定します。「true」に設定した場合、クエリーにはサーバ ID が含まれ、そのサーバのデータだけが監視されます。「false」に設定した場合、クエリーにサーバ ID は含まれず、クラスタ全体が 1 つのエンティティに保たれます。新しいパラメータはクラスタのすべてのメンバーで同じ値に設定する必要があります。このパラメータのデフォルト値はありません。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.serviceResults.cache
サービス結果のキャッシュに使用するパブリックキャッシュの名前を指定します。watt.server.serviceResults.cache パラメータの値を指定しないと、Integration Server は watt.server.serviceResults.cacheManager パラメータにサービス結果のキャッシュとして指定されたキャッシュマネージャの ServiceResults キャッシュを使用します。watt.server.serviceResults.cacheManager パラメータのキャッシュマネージャに ServiceResults という名前のキャッシュが含まれていない場合、Integration Server は起動時にエラーをスローし、サービス結果をキャッシュしません。

注:
SoftwareAG.IS.Services システムキャッシュマネージャにある ServiceResults システムキャッシュをサービス結果のキャッシュとして使用するには、watt.server.serviceResults.cache または watt.server.serviceResults.cacheManager の値を指定しないでください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.serviceResults.cacheManager
サービス結果のキャッシュに使用するパブリックキャッシュを含むパブリックキャッシュマネージャの名前を指定します。このパラメータの値が指定されていない場合、Integration Server はサービス結果のキャッシュマネージャとして SoftwareAG.IS.Services システムキャッシュマネージャを使用します。SoftwareAG.IS.Services システムキャッシュに、watt.server.serviceResults キャッシュの値に一致する名前を持つキャッシュが含まれていない場合、Integration Server は起動時にエラーをスローし、サービス結果をキャッシュしません。

注:
SoftwareAG.IS.Services システムキャッシュマネージャにある ServiceResults システムキャッシュをサービス結果のキャッシュとして使用するには、

watt.server.serviceResults.cache または watt.server.serviceResults.cacheManager の値を指定しないでください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.serviceResults.copyOnRead
キャッシュが返すエレメントをどのようにコピーするかを指定します。デフォルト値「true」の場合、結果を値で返します。値が「false」である場合、結果を参照で返します。

watt.server.serviceResults.copyOnWrite
キャッシュが書き込むエレメントをどのようにコピーするかを指定します。デフォルト値「true」の場合、結果を値で返します。値が「false」である場合、結果を参照で返します。

watt.server.serviceUsageDSP.RefreshInterval
[**サーバ**] > [統計情報] ページのデータを Integration Server Administrator がリフレッシュする間隔を秒単位で指定します。デフォルトは 90 秒です。このパラメータの変更は、次回 [**サーバ**] > [統計情報] ページが再ロードされたときに有効になります。

watt.server.session.locale.ignore
pub.date* サービスまたは pub.datetime* サービスのデフォルトロケールがサーバロケールであるか、またはそのサービスを呼び出したクライアントによって使用されるセッションのロケールであるかを指定します。「true」に設定した場合、ロケール入力パラメータ値が指定されていない pub.date* サービスの実行時に、Integration Server はロケールパラメータの値としてサーバロケールを使用します。「false」に設定した場合、*locale* 入力パラメータ値が指定されていない pub.date* サービスまたは pub.datetime* サービスの実行時に、Integration Server はそのサービスを呼び出したクライアントによって使用されるセッションのロケールを使用します。デフォルトは「false」です。

watt.server.session.stateful.enableLimit
ステートフルセッションの制限機能を有効にするかどうかを制御します。このプロパティを「`true`」に設定した場合、Integration Server は、ステートフルセッションを最大許容数 (watt.server.session.stateful.max プロパティで指定) に達するまで必要に応じて作成できます。Integration Server Administrator の **[サーバ] > [統計情報]** ページでステートフルセッションの統計情報を表示できます。

watt.server.session.stateful.max
Integration Server で同時に存在できるステートフルセッションの数を指定します。最大数のステートフルセッションが使用されているときに、ユーザがサーバにアクセスしてステートフルサービスを実行しようとすると、サーバはその要求を拒否し、エラーを返します。

watt.server.session.stateful.max の値は、正の整数にする必要があります。値を指定しなかった場合、または watt.server.session.stateful.enableLimit プロパティを「`false`」に設定している場合、同時に存在できるセッションの最大数は Integration Server ライセンスによって決まります。

注:
Integration Server Administrator の [リソースの設定の編集] ページの [**ステートフルセッションの最大数**] フィールドを使用して、最大値を設定することをお勧めします。ステー

トフルセッションの最大制限を設定する方法の詳細については、128 ページの「サーバセッションの管理」を参照してください。

watt.server.session.stateful.warning
Integration Server で、使用可能なステートフルセッションが不足していることを示す警告を生成するしきい値を指定します。使用可能なステートフルセッションの割合がこのプロパティの値以下になると、Integration Server は次のサーバログメッセージを生成します。

デフォルトは 25% です。

注:
Integration Server Administrator の [リソースの設定の編集] ページの [**ステートフルセッションの制限を有効にする**] フィールドを使用して、しきい値を設定することをお勧めします。使用可能なステートフルセッション警告しきい値の設定の詳細については、128 ページの「サーバセッションの管理」を参照してください。

watt.server.session.sweeperInterval
セッションスイーパタスクの実行間隔 (ミリ秒単位) を指定します。パラメータはすべての受信および送信セッションに適用されます。デフォルトは 120,000 ミリ秒 (2 分) です。

注:
watt.server.clientTimeout の値が watt.server.session.sweeperInterval より小さい場合、Integration Server はセッションスイーパスケジュール済みタスクの間隔として watt.server.clientTimeout の値を使用します。つまり、Integration Server は、セッションスイーパシステムタスクをスケジュールするとき、watt.server.session.sweeperInterval または watt.server.clientTimeout に設定された時間値のうち短い方の値を使用します。watt.sever.clientTimeout の値は単位として分を使用し、watt.server.session.sweeperInterval はミリ秒を使用することに注意してください。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.setResponse.pre82Mode
入力パラメータ *responseString* も *responseBytes* も指定されないときに、pub.flow:setResponse サービスが廃止された入力パラメータを検索して使用する順序を指定します。

「true」に設定すると、pub.flow:setResponse サービスは、Integration Server 7.1x および 8.0x で利用可能だったものと同様の優先順序に従います。具体的には、廃止されたパラメータを *response*、*string*、*bytes* の順序で検索し、見つかった最初のパラメータの値を使用します。

「false」に設定すると、pub.flow:setResponse サービスは、Integration Server 8.2 以降で利用可能だったものと同様の優先順序に従います。具体的には、廃止されたパラメータを *string*、*bytes*、*response* の順序で検索し、見つかった最初のパラメータの値を使用します。

デフォルトは「false」です。

注:
Integration Server 7.1 で導入された Web サービス実装が Integration Server 10.4 以降は廃止されたため、このパラメータは廃止されました。Integration Server バージョン 7.1 で導入された Web サービス実装は 8.2 より前の互換モードで実行される Web サービス記述子を処理します。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.sftp.dateStampFmt
SFTP クライアントパブリックサービス (具体的には WmPublic パッケージの pub.client.sftp* サービス) で使用される日付形式を指定します。使用する日付形式の指定には、Java クラス `java.text.SimpleDateFormat` でサポートされている任意の形式を使用できます。たとえば、08-12-02 14:44:33:1235 という形式で日付を表示するには、「dd-MM-yy HH:mm:ss:SSSS」と指定します。watt.server.sftp.dateStampFmt プロパティのデフォルト形式は *yyyy-MM-dd HH:mm:ss z* です。

watt.server.smtpServer
サーバエラーのログ取得およびサービスエラーの電子メール通知に使用する SMTP サーバを指定します。デフォルトはありません。

watt.server.smtpServerPort
Integration Server がサーバエラーのログ取得およびサービスエラーの電子メール通知の送信先とする、SMTP サーバ上の受信待機中ポート数を指定します。デフォルトは 25 です。

watt.server.smtp.userName
Integration Server が SMTP サーバに接続するときに使用するアカウントのユーザ名を指定します。

watt.server.smtpTransportSecurity
Integration Server が SMTP サーバと通信するときに使用する SSL 暗号化のタイプを指定します。
このプロパティは、次のいずれかの値に設定します。

設定値	Integration Server
なし	SMTP サーバとの通信時にセキュアでないモードを使用します。 「**なし**」を指定した場合、SMTP サーバはユーザ名とパスワードだけを使用して SMTP クライアントを認証します。
[**明示的**]	SMTP サーバとの通信時に明示的なセキュリティを使用します。明示的なセキュリティを指定した場合、Integration Server は SMTP サーバへの接続を暗号化なしで確立してから、セキュアモードにアップグレードします。
[**暗黙的**]	SMTP サーバとの通信時に暗黙的なセキュリティを使用します。暗黙的なセキュリティを指定した場合、Integration Server は SMTP サーバへの接続を常に暗号化付きで確立します。SSL をサポートしているクライアントのみアクセスを許可されます。

watt.server.smtpTrustStoreAlias
Integration Server で Integration Server と SMTP サーバ間の信用関係の妥当性を検査するときに使用する認証のリストを含むトラストストアのエイリアスを指定します。トラストストアエイリアスを指定しなかった場合、watt.security.trustStoreAlias プロパティで指定したデフォルトのトラストストアエイリアスが使用されます。トラストストアエイリアスの詳細に

ついては、512ページの「キーストア、トラストストアおよび鍵のエイリアス」を参照してください。

watt.server.SOAP.addEmptyHeader
Integration Server で、pub.soap.utils:createSoapData または pub.soap.utils:stringToSoapData サービスを実行することによって作成された SOAP メッセージ内に空のヘッダーエントリを作成するかどうかを指定します。このプロパティを「true」に設定した場合、Integration Server は SOAP メッセージ内に空のヘッダーを作成します。このプロパティを「false」に設定した場合、Integration Server は SOAP メッセージ内にヘッダーエントリを作成しません。デフォルトは「true」です。

pub.soap.utils:createSoapData サービスまたは pub.soap.utils:stringToSoapData サービスの *addEmptyHeader* パラメータを設定した場合は、その値が watt.server.SOAP.addEmptyHeader の設定よりも優先されます。

pub.soap.utils:createSoapData および pub.soap.utils:stringToSoapData サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.SOAP.completeLoad
pub.utils:addBodyEntry、pub.utils:addHeaderEntry、pub.utils:addTrailer によって追加された XML ノードを SOAP メッセージへの追加前に完全にロードするかどうかを指定します。「true」に設定した場合、Integration Server は XML ノード全体をロードします。「false」に設定した場合、Integration Server は XML ノードをサービスへの追加前に完全にはロードしません。デフォルトは「true」です。

watt.server.soap.convertPlainTextHTTPResponseIntoSOAPFault
Integration Server がクライアントとして機能しており、Web サービスがプレーンテキスト HTTP 応答を生成する場合に、Web サービス呼び出しからの応答を返す方法を指定します。このパラメータは、次のバージョンを使用して Web サービスコネクタが作成された場合にのみ適用されます。

- Integration Server バージョン 8.2 以上 (ただし、Web サービス記述子の [**8.2 より前のバージョンとの互換モード**] プロパティが「true」の場合)
- Integration Server バージョン 8.2 より前のバージョン

watt.server.soap.convertPlainTextHTTPResponseIntoSOAPFault パラメータを「true」 (デフォルト) に設定した場合、Integration Server はプレーンテキスト HTTP 応答を変換し、Web サービスコネクタの出力パラメータ内で HTTP 応答からの情報を返します。

- Integration Server バージョン 8.2 以上を使用して Web サービスコネクタが作成され、[**8.2 より前のバージョンとの互換モード**] プロパティが「true」の場合、Integration Server は Web サービスコネクタの *fault/reasons* 出力パラメータで変換後の HTTP ペイロードを返します。
- Integration Server バージョン 8.2 より前のバージョンを使用して Web サービスコネクタが作成された場合、Integration Server は使用中の SOAP プロトコルに基づいて次のいずれかの方法で変換後の HTTP ペイロードを返します。
  - *SOAP-FAULT/Fault_1_1/faultstring*
  - *SOAP-FAULT/Fault_1_2/SOAP-ENV:Reason/SOAP-ENV:Text*

プレーンテキスト HTTP 応答を変換し、Web サービスコネクタの出力パラメータで返すと、例外の原因の特定に役立つことがあります。

このパラメータを「false」に設定した場合、Web サービスがプレーンテキスト HTTP 応答を生成すると、Integration Server は無効な SOAP エンベロープを受信したことを示す例外をスローします。

注:
Integration Server 7.1 で導入された Web サービス実装が Integration Server 10.4 以降は廃止されたため、このパラメータは廃止されました。Integration Server バージョン 7.1 で導入された Web サービス実装は 8.2 より前の互換モードで実行される Web サービス記述子を処理します。

重要:
このパラメータを更新した場合、その変更を適用するには、Integration Server を再起動する必要があります。

watt.server.soap.decodeElementWithPrefix
定義済み XML ネームスペース URI があり、各ネームスペースとプリフィックスが関連付けられていないドキュメントタイプについて、Integration Server が認識するかどうかを指定します。デフォルトで、Integration Server バージョン 8.2 SP2 以上では、定義済み XML ネームスペース URI があり、各ネームスペースとプリフィックスが関連付けられていないドキュメントタイプについて、サービスへの入力として指定されている場合、SOAP プロセッサは実行時にドキュメントタイプを認識できません。watt.server.soap.decodeElementWithPrefix プロパティが「true」に設定されると、定義済み XML ネームスペース URI があり、各ネームスペースとプリフィックスが関連付けられていないドキュメントタイプについて、Integration Server は認識します。

デフォルトは「false」です。

watt.server.soap.decodeElementWithPrefix サーバ設定パラメータの値は、Integration Server が Integration Server バージョン 7.1 で作成された Web サービス記述子用に受信 SOAP メッセージをデコードする方法に影響を与えます。具体的には、このパラメータは、Integration Server が 7.1.x Web サービスコネクタによって受信される SOAP 応答メッセージと 7.1.x サービスプロバイダによって受信される SOAP 応答メッセージをデコードする方法に影響を与えます。

- watt.server.soap.decodeElementWithPrefix を「true」に設定すると、Integration Server は着信 SOAP メッセージをデコードするときに、7.1.x で作成される Web サービス記述子の一部である Web サービスまたは Web サービスコネクタ用に SOAP メッセージを IData に デコードする場合、ネームスペースやプリフィックス記述子を考慮しません。Integration Server は、フィールド名にプリフィックスがない場合でも、XML 文字列のネームスペースで完全修飾された要素を IS ドキュメントタイプのフィールドにマッピングします。つまり、パラメータを「true」に設定すると、Integration Server は、Integration Server 7.1.x で作成された Web サービス記述子の一部である Web サービスまたは Web サービスコネクタに対して、バージョン 7.1.x で使用できるデコーディング動作を使用します。

  注:
  上で説明した Integration Server 7.1.x デコーディング動作は無効になります。ただし、Integration Server 8.2 以降の厳密なデコーディングプロセスでは、フィールドが見つからないか過剰になり、検証エラーになります。
  watt.server.soap.decodeElementWithPrefix を「true」に設定すると、7.1.x の動作が可能になり、7.1.x Web サービス記述子の変更は不要です。

- watt.server.soap.decodeElementWithPrefix を「false」に設定すると、Integration Server は着信 SOAP メッセージをデコードするときに、Integration Server 7.1.x で作成される Web サービス記述子の一部である Web サービスまたは Web サービスコネクタの IData に SOAP メッセージをデコードする場合、ネームスペースやプリフィックス記述子を考慮します。フィールド名にプリフィックスが含まれない場合、Integration Server は、XML 文字列内のネームスペースで完全修飾された要素を IS ドキュメントタイプのフィールドにマッピングしません。つまり、パラメータを「false」に設定すると、Integration Server は、Integration Server 7.1.x 以降で作成された Web サービス記述子の一部である Web サービスまたは Web サービスコネクタに対して、バージョン 8.2 SP2 以降で使用できるデコーディング動作を使用します。

注:
Integration Server 8.2 SP2 以降で作成された Web サービス記述子では、プリフィックスをサービスの署名内のフィールドの XML ネームスペース URI と関連付ける必要があります。

また、watt.server.soap.decodeElementWithPrefix サーバ設定パラメータは、pub.xml.xmlNodeToDocument サービスがネームスペースで完全修飾された要素を IS ドキュメントタイプの対応するフィールドにデコードするときに名前スペースやプリフィックス宣言を考慮するかどうかも判断します。

- watt.server.soap.decodeElementWithPrefix を「true」に設定すると、pub.xml:xmlNodeToDocument サービスはネームスペースやプリフィックス宣言を考慮しません。サービスは、フィールド名にプリフィックスがない場合でも、XML 文字列のネームスペースで完全修飾された要素を IS ドキュメントタイプのフィールドにマッピングします。この動作は無効であると判断されますが、以前のバージョンの Integration Server で展開されるサービスとの下位互換性を提供できます。

- watt.server.soap.decodeElementWithPrefix を「false」に設定すると、pub.xml:xmlNodeToDocument サービスは、ネームスペース/プリフィックス宣言を考慮します。フィールド名にプリフィックスが含まれない場合、サービスは、XML 入力内のネームスペースで完全修飾された要素を IS ドキュメントタイプのフィールドにマッピングしません。pub.xml:xmlNodeToDocument サービスは、プリフィックスなしのフィールドは生成しません。その代りに、サービスは prefix:name 構造を使用する新しいフィールドを追加し、XML インスタンスの値からこれらのフィールドに値を入力します。

重要:
watt.server.soap.decodeElementWithPrefix パラメータの更新後、パラメータの変更を有効にするには、Integration Server を再起動する必要があります。

watt.server.SOAP.default.endpointHTTP

HTTP プロトコルのデフォルトプロバイダ Web サービスエンドポイントエイリアスを指定します。このパラメータの値が空である場合、HTTP のデフォルトプロバイダ Web サービスエンドポイントエイリアスはありません。

このパラメータのデフォルト値はありません。

watt.server.SOAP.default.endpointHTTP パラメータは、Integration Server Administrator の [**設定**] > [Web サービス] > [デフォルトプロバイダエンドポイントエイリアスの設定] ページで [**HTTP**] パラメータに指定される値を表示します。デフォルトプロバイダエンドポイントエイリアスの設定の詳細については、377 ページの「プロバイダ Web サービス記述子のデフォルトエンドポイントエイリアスの設定」を参照してください。

watt.server.SOAP.default.endpointHTTPS
HTTPS プロトコルのデフォルトプロバイダ Web サービスエンドポイントエイリアスを指定します。このパラメータの値が空である場合、HTTPS のデフォルトプロバイダ Web サービスエンドポイントエイリアスはありません。

このパラメータのデフォルト値はありません。

watt.server.SOAP.default.endpointHTTPS パラメータは、Integration Server Administrator の **[設定] > [Web サービス] > [デフォルトプロバイダエンドポイントエイリアスの設定]** ページで **[HTTPS]** パラメータに指定される値を表示します。デフォルトプロバイダエンドポイントエイリアスの設定の詳細については、377 ページの「プロバイダ Web サービス記述子のデフォルトエンドポイントエイリアスの設定」を参照してください。

watt.server.SOAP.defaultProtocol
新しい SOAP メッセージに関して Integration Server で使用するデフォルトプロトコルを指定します。SOAP 1.1 プロトコルまたは SOAP 1.2 プロトコルを指定してください。デフォルトは SOAP 1.1 プロトコルです。

watt.server.SOAP.directive
要求を Integration Server SOAP ハンドラにルーティングする URL の SOAP ディレクティブに使用する別の単語を指定します。デフォルトでは、このパラメータは `watt.server.SOAP.directory=soap` と設定され、ユーザは SOAP ディレクティブを `soap` (`http://`*`host`*`:`*`port`*`/soap`) と指定する必要があります。soap ではなく別の単語で SOAP ディレクティブを指定することをユーザに許可する場合は、このパラメータに別の単語を設定します。たとえば、SOAP ディレクティブを `endpoint` と指定することをユーザに許可する場合は (`http://`*`host`*`:`*`port`*`/endpoint`)、このパラメータを `watt.server.SOAP.directive=endpoint` と設定します。

watt.server.SOAP.encodeXSIType
RPC/Encoded を指定するエレメントの属性として xsi:type を SOAP メッセージに含めるかどうかを指定します。値「true」を指定すると、Integration Server はエレメントの属性として xsi:type を含めます。「false」を指定すると、xsi:type 属性は省略されます。デフォルトは「true」です。Software AG では、このパラメータの値として「true」を使用することをお勧めします。

watt.server.SOAP.encodeXSITypeValue
SOAP の要求と応答で、Integration Server に xsi:type 属性と xsd:anyType エレメントの値を含めるかどうかを示します。「false」に設定した場合、Integration Server は SOAP 要求と応答で xsi:type 属性と xsd:anyType エレメントの値を省略します。「true」に設定した場合、Integration Server は SOAP 要求と応答で xsi:type 属性と xsd:anyType エレメントの値を含みます。デフォルトは「true」です。

> **注:**
> watt.server.SOAP.encodeXSITypeValue サーバ設定パラメータは、Integration Server で作成されたこれらの SOAP 要求と応答のみに影響を及ぼします。

watt.server.SOAP.generateNilTags
Integration Server で SOAP メッセージのエレメントの属性として xsi:nil を生成するかどうかを指定します。「true」に設定した場合、Integration Server は NULL 値可のエレメント (対応するフィールドの **[NULL値を許可]** プロパティが「true」に設定されていて、実行時にフィールドが NULL になるエレメント) に対して xsi:nil 属性を生成します。
watt.server.SOAP.generateNilTags を「false」に設定した場合、対応するフィールドが NULL

値可で、実行時に NULL になるときでも、Integration Server はそのエレメントの xsi:nil 属性を省略します。デフォルトは「true」です。

watt.server.SOAP.generateRequiredTags
実行時に値が指定されなかった必須パラメータの空エレメントタグを Integration Server によって生成される SOAP メッセージに含めるかどうかを指定します。「true」に設定した場合、Integration Server は値が指定されなかった必須パラメータの空エレメントタグを生成します。watt.server.SOAP.generateRequiredTags を「false」に設定した場合、Integration Server は値が指定されなかった必須パラメータの空エレメントタグを省略します。デフォルトは「true」です。

watt.server.SOAP.hideEPRHostInFault
SOAP の障害において、エンドポイント参照のホスト名および IP アドレスの詳細を隠します。このパラメータを「true」に設定した場合、Integration Server は SOAP の障害におけるホスト名および IP アドレスを *:* に置き換えます。このパラメータを「false」に設定した場合、Integration Server の SOAP の障害には、エンドポイント参照のホスト名および IP アドレスの詳細が含まれます。デフォルトは「false」です。

注:
このパラメータは、Web サービス記述子の [**8.2 より前のバージョンとの互換モード**] プロパティが「false」に設定されている場合だけ適用されます。

watt.server.SOAP.hideFaultReason
このパラメータを使用すると、Web サービス要求への応答から障害の詳細を削除できます。デフォルトで、このパラメータは「false」に設定され、応答には障害の詳細が含まれます。応答に障害の詳細が含まれる場合、クライアントアプリケーションは Web ページで実際のコンテンツを偽造または変更する可能性があります。コンテンツの偽造を防ぐには、このパラメータを「true」に設定します。

重要:
また、Web サービスがエラーを返した場合に Integration Server がスタックトレースをクライアントに返さないようにするには、watt.server.http.returnException を「false」または「webMethods」に設定する必要があります。

watt.server.SOAP.identifyIsGeneratedWSDL
WSDL ドキュメントから Web サービス記述子を作成または更新するときに、WSDL ドキュメントが Integration Server によって生成されたかどうかを検知します。
watt.server.SOAP.identifyISGeneratedWSDL プロパティを「true」に設定すると、Integration Server は WSDL ドキュメントが Integration Server によって生成されたかどうかを検知し、その場合、入力および出力メッセージのトップレベルエレメントを削除します。「false」に設定すると、Web サービス記述子の作成時に Integration Server は WSDL ドキュメントを書き込み済みとして処理し、入力および出力メッセージのトップレベルのエレメントを削除しません。デフォルト値は true です。

このパラメータは WSDL の最初のプロバイダ Web サービス記述子およびコンシューマ Web サービス記述子に影響します。

重要:
watt.server.SOAP.identifyISGeneratedWSDL の値の変更は、Web サービス記述子の作成と更新のロジックに影響します。パラメータの値を変更し、その後パラメータの値が変更さ

れる前に作成されたWebサービス記述子を更新する場合、更新されたWebサービス記述子は元の Web サービス記述子とは異なります。

watt.server.SOAP.ignoreMissingResponseHeader
SOAP の障害を含め、SOAP 応答に必要なヘッダーがない場合にエラーにするかどうかを決定します。`true` に設定すると、SOAP 応答に必要がヘッダーがない場合でもエラーが返されません。`false` に設定すると、SOAP 応答に必要がヘッダーがない場合はエラーになります。デフォルトは「`false`」です。

Webサービス記述子のすべてのヘッダーは、元のWSDLドキュメントから生成されたものでも、Webサービス記述子に追加されたものでも、必須として処理されます。これはWSDLのすべてのヘッダーが必須として処理されると宣言しているWS-I基本プロファイルのルールを採用したものです。バージョン 9.0 以前の Integration Server では、Web サービス記述子により受信された SOAP 応答の処理で、すべての SOAP ヘッダーが存在するかどうかが適切に検証されませんでした。Integration Serverその結果、SOAP応答メッセージにSOAPヘッダーがない場合にIntegration Serverは、エラーをスローしませんでした。Integration Server バージョン 9.0 より、Integration Server は SOAP 応答の必要なヘッダーを適切に検証します。必要なヘッダーが存在しない場合は、Integration Serverはエラー「[ISS.0088.9443] SOAP応答に 1 つ以上の必要なヘッダー <headerName> が存在しません」をスローします。必要なヘッダーがないときにエラーが正しい動作である一方、Integration Server は SOAP 応答に必要なヘッダーがない場合にエラーにするかどうかを制御する設定プロパティを提供しています。これはマイグレーションを行うときに有用です。

watt.server.SOAP.inbound.CDATA.removeTags
Integration Server が受信 SOAP 要求で見つかった CDATA 区切りタグを削除するか、または維持するかを指定します。「true」に設定すると、受信処理中に Integration Server は初期 CDATA タグおよび終了タグをターゲット Web サービスへの入力として渡されたストリング値から削除します。「false」に設定すると、Integration Server は CDATA 区切りタグを受信 SOAP 要求に維持し、CDATA 区切りタグは要求のテキスト内に残り、ターゲット Web サービスに到達します。デフォルト値は true です。

watt.server.SOAP.MTOMStreaming.cachedFiles.location
Integration Server で、MTOM ストリーミングの実行時に受信 SOAP メッセージを一時的に格納するために使用するハードディスクドライブ領域への絶対パスを指定します。このパラメータは、MTOM ストリーミングが有効になっている場合 (つまり、watt.server.SOAP.MTOMStreaming.enable が「true」に設定されている場合) にのみ適用されます。

Integration Server と同じマシン上のディレクトリを指定することも、Integration Server からアクセス可能な他の場所のディレクトリ (マッピングされた論理ドライブやネットワークディレクトリなど) を指定することもできます。デフォルト値は *Integration Server_directory* \instances*instance_name*\temp\mtom\cached files です。

注:
Integration Server をクラスタ環境で使用している場合、クラスタ内のすべてのサーバでこのプロパティを設定する必要があります。Integration Server のクラスタ設定の詳細については、『*webMethods Integration Server Clustering Guide*』を参照してください。

watt.server.SOAP.MTOMStreaming.enable
受信 SOAP メッセージに対して MTOM ストリーミングが有効であるかどうか、送信要求に対して HTTP チャンクが有効であるかどうかを指定します。このプロパティを「true」に設定

した場合、受信 SOAP メッセージに対する MTOM ストリーミングおよび送信要求に対する HTTP チャンクを有効化します。このプロパティを「false」に設定した場合、MTOM ストリーミングおよび HTTP チャンクを無効化します。デフォルトは「`false`」です。

watt.server.SOAP.MTOMStreaming.threshold
Integration Server が受信 MTOM 添付フィールドをメモリ内 ByteStream として処理する最大バイト数を指定します。Integration Server はこのサイズを超える受信 MTOM 添付フィールドを一時ディスクファイルに書き込み、受信 MTOM ストリームを FileStream として処理します。このパラメータは、MTOM ストリーミングが有効になっている場合 (watt.server.SOAP.MTOMStreaming.enable が「true」に設定されている場合) にのみ適用されます。デフォルトは 4000 バイトです。

watt.server.SOAP.MTOMStreaming.cachedFiles.location 値は一時ディスクファイルの場所を決定します。

watt.server.SOAP.MTOMThreshold
フィールドサイズ (KB 単位) を指定します。これによって、送信 SOAP 要求に含まれる base64binary エンコードデータを Integration Server が MIME 添付データとして扱うか、SOAP メッセージ内に含めて送信するかが決まります。SOAP メッセージの Web サービス記述子によって SOAP 要求へのデータ添付が許可されている場合、Integration Server は、指定されたしきい値を超える、SOAP メッセージ内の base64 フィールドすべてを MIME 添付データとして送信します。デフォルトは 0 です。

watt.server.SOAP.pre82WSD.ignoreVersionMismatch
8.2.2 より前の Integration Server リリースで作成され、[**8.2 より前のバージョンとの互換モード**] プロパティが「true」に設定されている Web サービスプロバイダに対して、プロセス SOAP 要求に対して 8.2 より前のバージョンの動作をエミュレートするかどうかを指定します。Integration Server 7.1.3 では、Web サービスプロバイダが WSDL バインディングで宣言された SOAP バージョンに一致しない SOAP バージョンで SOAP 要求を処理することができました。このプロパティを「true」に設定すると、コンシューマが SOAP 要求を更新する必要がないように下位互換性が提供されます。

「false」に設定した場合、Integration Server は妥当性検査をより厳密にし、この不一致が存在するときに処理を許可しません。デフォルト値は「false」です。

> **注:**
> Integration Server 7.1 で導入された Web サービス実装が Integration Server 10.4 以降は廃止されたため、このパラメータは廃止されました。Integration Server バージョン 7.1 で導入された Web サービス実装は 8.2 より前の互換モードで実行される Web サービス記述子を処理します。

watt.server.SOAP.preserveCDATA
Integration Server が送信メッセージで CDATA ブロックをエンコードするかどうかを指定します。「true」に設定した場合、Integration Server の送信 SOAP メッセージに CDATA があると、Integration Server はワイヤ要求で CDATA を未変更でエンコードせずに保持します。「false」に設定した場合、Integration Server では CDATA セクションを通常のテキストとして扱うため、CDATA セクションの CDATA タグと不正な文字は HTML エンコードされます。Integration Server がワイヤに送信 SOAP メッセージを載せる前にエンコードが実行されることに注意してください。この設定の影響を受ける送信 SOAP メッセージには、Web サービスコネクタによって送信される SOAP 要求、および Web サービスプロバイダによって送信される SOAP 応答が含まれます。デフォルト値は「true」です。CDATA は保持され、エンコードされません。

watt.server.SOAP.request.timeout
リモートプロシージャをホストするサーバからの SOAP 応答を Integration Server で待機する時間を秒単位で指定します。Integration Server で割り当て時間内に応答を受信しなければ、要求は終了されます。デフォルト値は -1 で、Integration Server が watt.net.timeout プロパティに設定された値を使用することを示します。

watt.server.SOAP.retainUndeclaredNamespace
Integration Server が、SOAP 要求または SOAP 応答のデコード時に xsd:any エレメントからのネームスペースを保持するかどうかを指定します。指定可能なエレメントと対応するネームスペースが対応するドキュメントタイプに定義されていないため、Integration Server は xsd:any エレメントからのネームスペースを宣言されていないと見なします。「true」に設定した場合、Integration Server は xsd:any エレメントのネームスペース宣言を保持します。さらに、ネームスペースに属するエレメントの対応するフィールドのフィールド名にはプリフィックスが含まれます。「false」に設定した場合、Integration Server は xsd:any エレメントのネームスペース宣言を保持せず、宣言されたネームスペース名のエレメントに対応するフィールド名にプリフィックスは含まれません。デフォルトは「true」です。

このパラメータを「true」に設定した場合、Integration Server は、xsd:any エレメントの各ネームスペース宣言のドキュメントに次のフィールドを挿入することで、ネームスペース宣言を保持します。

@xmlns:<prefix>

<prefix> は SOAP メッセージで定義されたプリフィックスです。プリフィックスが定義されていない場合、変数名は @xmlns になります。

> **注:**
> watt.server.SOAP.retainUndeclaredNamespace の値は Web サービスのすべての SOAP メッセージのデコードに影響します。

watt.server.SOAP.setNamespaceURIsToRoot
Integration Server が SOAP 応答で XML ネームスペースを宣言する方法を指定します。watt.server.SOAP.setNamespaceURIsToRoot プロパティが「true」に設定されると、Integration Server は SOAP 応答のルートエレメントでネームスペースおよびプリフィックスを 1 回宣言してから、応答内の各エレメントでプリフィックスを使用します。「false」に設定すると、Integration Server は元のドキュメントで定義されているとおりに、完全なネームスペースを使用して明示的に各エレメントを定義するか、またはネームスペースおよびプリフィックスをルートエレメントで 1 回だけ宣言するかして、ネームスペースを宣言します。このプロパティのデフォルト値は「`false`」です。

watt.server.SOAP.streamHandlers
ストリーム入力を想定するカスタム SOAP ハンドラを指定します。SOAP ハンドラは、セミコロン (;) 区切りリストで入力します。デフォルトはありません。

watt.server.SOAP.treatNilAsNull
Integration Server が SOAP メッセージのデコーディング時に、xsi:nil 属性が true に設定されている要素で @xsi:nil フィールドを生成するかどうかを示します。`true` に設定すると、Integration Server は、xsi:nil 属性を伝達する要素を null 値として扱い、要素の @xsi:nil 属性を作成しません。このパラメータを `false` に設定すると、Integration Server はxsi:nil=true 属性を伝達する要素の @xsi:nil フィールドを生成します。この設定パラメータは、Integration Server で実行されるすべての Web サービスに影響します。デフォルトは「`true`」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.SOAP.useMultiReference
SOAP メッセージ内の複数の場所に存在するエレメントに対して、Integration Server で全出現箇所のエレメントをシリアライズするか、1 箇所でのみシリアライズして他の箇所では href 属性を使用するかを指定します。「true」に設定した場合、Integration Server は 1 箇所をシリアライズして、他の箇所では href 属性を使用します。このパラメータは、RPC/Encoded Web サービスにのみ適用されます。デフォルトは「`true`」です。

watt.server.SOAP.useStringforAnyTypewithSimpleValue
SOAP 応答内の単純なコンテンツを持つ xsd:anyType エレメントを Integration Server がデコードする方法を指定します。「`false`」に設定した場合、Integration Server は単純なコンテンツを持つ xsd:anyType エレメントを、@xsi:type フィールドを持つ IData にデコードして、xsi:type 値と *body フィールドを保持し、エレメント値を含めます。これがデフォルトの動作です。

watt.server.SOAP.useStringforAnyTypewithSimpleValue サーバ設定パラメータを「`true`」に設定した場合、Integration Server は単純なコンテンツを持つ xsd:anyType エレメントを、エレメント値を含む String タイプフィールドにデコードします。エレメントの xsi:type 情報は失われます。変更した内容を有効にするために Integration Server を再起動する必要はありません。

watt.server.soap.validateInput
受信 SOAP 要求にサービス入力署名に宣言されていないフィールドが含まれているときに妥当性検査エラーを発生させるかどうかを指定します。このパラメータを「true」に設定すると、受信 SOAP 要求の本文にサービス入力署名で宣言されていないフィールドが含まれているときに Integration Server は妥当性検査エラーを返します。このパラメータを「false」に設定すると、SOAP 本文にサービス入力署名に存在しないフィールドが含まれていても Integration Server は受信 SOAP 要求の処理を続行します。デフォルトは「true」です。

watt.server.soap.validateResponse
SOAP 応答の妥当性検査を有効または無効にします。「`true`」に設定した場合、Integration Server は Web サービスコネクタで受信した SOAP 応答の妥当性検査を行います。「`false`」に設定した場合、Integration Server は受信した SOAP 応答の妥当性検査を行いません。デフォルトは「`true`」です。

watt.server.SOAP.validateSOAPMessage
Integration Server が Web サービスプロバイダとして機能している場合に、Integration Server で受信 SOAP 要求および送信 SOAP 応答の妥当性検査を行うかどうかを指定します。「`true`」に設定した場合、Integration Server は SOAP スキーマに対して受信 SOAP 要求および送信 SOAP 応答の妥当性検査を行います。妥当性検査プロセスでは、メッセージエンベロープが正しく構成されているかどうかをチェックするだけであることに注意してください。たとえば、Integration Server は、メッセージに 1 つ以上の本文要素があり、最大 1 つのヘッダー要素があることをチェックします。Integration Server はメッセージによって保持されるデータの妥当性検査は行いません。デフォルトは「`true`」です。

使用中の実稼動環境で、サーバにサブミットされたメッセージの妥当性が確認されている場合、watt.server.SOAP.validateSOAPMessage を「false」に設定してパフォーマンスを最適化することができます。

watt.server.SOAP.warnOnPartValidation

WSDL ドキュメントから Web サービス記述子を作成するときに、Integration Server がエレメント属性の代わりにタイプ属性で定義されるメッセージパートをエラーではなく警告として扱うかどうかを指定します。このパラメータを「true」に設定すると、Integration Server は警告を返し、Web サービス記述子の作成を許可することを示します。このパラメータを「false」に設定すると、Integration Server はエラーを返し、Web サービス記述子の作成を許可しないことを示します。デフォルトは「false」です。

watt.server.soapJMS.defaultMessageType
JMS を介して SOAP で送信される Web サービス要求メッセージのデフォルトメッセージタイプを指定します。本文に中断のないバイトストリームが含まれるメッセージを送信するには、このパラメータを「BytesMessage」に設定します。本文に Java ストリングが含まれるメッセージを送信するには、このパラメータを「TextMessage」に設定します。「BytesMessage」は、JMS メッセージを効率よく送信する方法として考えられています。ただし、デバッグの目的では、必要に応じてパラメータを「TextMessage」に設定して、人間が読み取ることができる形式のメッセージを送信できます。要求メッセージのメッセージタイプによって応答メッセージのメッセージタイプが決まることに留意してください。デフォルトは「BytesMessage」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

注:
transportHeaders 入力パラメータの jms.messageType プロパティを設定することによって、Web サービスコネクタの実行中にデフォルトメッセージタイプを上書きできます。

注:
このパラメータを TextMessage に設定すると、Integration Server はすべての Web サービス応答を TextMessage として送信します。要求メッセージが BytesMessage で、watt.server.soapJMS.defaultMessageType が TextMessage に設定されている場合、Integration Server は要求メッセージタイプを上書きし、応答を TextMessage として送信します。これはデバッグを行うときに有用です。

watt.server.soapjms.request.timeout
JMS を介して SOAP で送信される SOAP 要求への応答を Integration Server が待機する秒数を指定します。この値は、ゼロ以上の整数にする必要があります。値 0 は、Integration Server が応答を無期限に待機することを示します。デフォルトは 10 秒です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.SoapRPC.checkHeaders
Integration Server が SOAP RPC 要求の SOAP ヘッダーをチェックするかどうかを示します。デフォルトは「true」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.SoapRPC.distinguishDuplicateElements
SOAP/RPC Web サービスの SOAP 応答で、同じ名前の配列を Integration Server が区別する必要があるかどうかを指定します。「true」に設定した場合、Integration Server は xsi:type

値に番号を付加し、SOAP 応答内の同じ名前の配列エレメントを区別します (`elementName` および `elementName1` など)。「`false`」に設定した場合、Integration Server は SOAP 応答内の同じ名前の配列を区別しません。デフォルトは「`true`」です。

watt.server.SoapRPC.useSecondaryType
入力署名または出力署名に異なるタイプの同名の変数を使用しているサービスに対して SOAP 応答を作成するときに、2 つ目の型定義を使用するように Integration Server に指示します。異なるタイプの同名のフィールドを使用しているサービスを含むプロバイダ Web サービス記述子から WSDL を作成する場合、Integration Server は WSDL でそのフィールドタイプの 2 つ目のインスタンスの名前を変更します。RPC-Encoded SOAP バインディングを使用している場合、Integration Server は実行時に SOAP 応答内のタイプをエンコードします。このプロパティを「`true`」に設定した場合、SOAP 応答は、名前が変更された型定義を参照します。「`false`」に設定した場合、SOAP 応答は、名前が変更された型定義ではなく元の型定義を参照します。デフォルトは「`false`」です。このプロパティは、RPC-Encoded SOAP バインディングにのみ適用されます。

watt.server.ssl.keyStoreAlias
SSL 対応外部サーバとの SSL 接続を確立するのに必要な情報を含む、Integration Server キーストアのキーストアエイリアス名。このパラメータのデフォルト値はありません。外部サーバへの SSL 接続に関するキーストア情報の保存の詳細については、498 ページの「Integration Server JVM の SSL 情報の設定」を参照してください。

watt.server.ssl.keyStoreAlias パラメータにより、Integration Server Administrator の **[セキュリティ] > [認証]** ページおよび **[セキュリティ] > [認証] > [認証設定の編集]** ページの [**JVM キーストアエイリアス**] フィールドに指定された値が表示されます。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

重要:
キーストアエイリアスを watt.server.ssl.keyStoreAlias パラメータの値として使用しているキーストアのコンテンツを変更した場合は、Integration Server を再起動する必要があります。キーストアに対する変更は再起動後に反映されます。

watt.server.ssl.trustStoreAlias
SSL 対応外部サーバとの SSL 接続を確立するのに必要な情報を含む、Integration Server トラストストアのトラストストアエイリアス名。このパラメータのデフォルト値はありません。外部サーバへの SSL 接続に関するトラストストア情報の保存の詳細については、498 ページの「Integration Server JVM の SSL 情報の設定」を参照してください。

watt.server.ssl.trustStoreAlias パラメータにより、Integration Server Administrator の **[セキュリティ] > [認証]** ページおよび **[セキュリティ] > [認証] > [認証設定の編集]** ページの [**JVM トラストストアエイリアス**] フィールドに指定された値が表示されます。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

重要:
トラストストアエイリアスを watt.server.ssl.trustStoreAlais パラメータの値として使用しているトラストストアのコンテンツを変更した場合は、Integration Server を再起動する必要があります。トラストストアに対する変更は再起動後に反映されます。

watt.server.stats.clearStatsOnLogRotate
サーバログまたは stats ログが循環する場合に、Integration Server がサービスエラーメトリクスおよびシステムエラーメトリクスを消去するかどうかを指定します。サービスエラー数は、[サーバ] > [統計情報] ページの [要求数] テーブルの現在の [**サービスエラー**] フィールドおよび Prometheus メトリック `sag_is_number_service_errors` で使用されます。システムエラー数は Prometheus メトリック `sag_is_number_nonservice_errors` で使用されます。サーバログまたは stats ログの循環時に、サービスエラーおよびシステムエラーメトリクスをリセットするには、「true」に設定します。ログの循環時にサービスエラーおよびシステムエラーメトリクスを消去しないことを示すには、「false」に設定します。デフォルトは「false」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.stats.logfile
Integration Server で統計ログ情報を書き込む先のファイルの完全修飾パスまたは相対パスを指定します。ファイルには、[**サーバ**] > [統計情報] ページに表示されるデータが含まれます。指定するパスは存在している必要があります。存在していないパスを指定すると、Integration Server は開始しません。ファイルが存在していない場合は、Integration Server は最初に統計ログを書き込むときにファイルを作成します。相対パスは Integration Server ホームディレクトリ *Integration Server_directory* \instances*instance_name* に相対的なパスです。

デフォルトは `logs\stats.csv` です。

watt.server.stats.logfle に設定する統計ログのファイルタイプは、watt.server.stats.logfile.csv server 設定パラメータの値と整合性があるファイルタイプに設定してください。例えば、watt.server.stats.logfile.csv が「true」に設定されている場合、統計ログファイルの拡張子は csv である必要があります。watt.server.stats.logfile.csv は「false」に設定されている場合は、統計ログは 16 進数形式になり、統計ログファイルの拡張子は .log になります。

起動時に、watt.server.stats.logfile.csv が「true」に設定されていて、watt.server.stats.logfile が logs/stats.log に設定されている場合、Integration Server は、watt.server.stats.logfile の値を logs/stats.csv に変更します。起動時に、watt.server.stats.logfile.csv が「false」に設定されていて、watt.server.stats.logfile が logs/stats.csv に設定されている場合、Integration Server は、watt.server.stats.logfile の値を logs/stats.log に変更します。

watt.server.stats.logfile.csv
統計ログの形式を指定します。「true」に設定されている場合は、Integration Server は統計ログを stats.csv という名前のカンマ区切りファイルに書き込みます。「false」に設定されている場合は、Integration Server は統計ログを stats.log という名前の 16 進数形式ファイルに書き込みます。デフォルトは「true」です。

watt.server.stats.logfile.csv を「false」に設定する場合、watt.server.stats.logfile の値を *.log ファイルに変更していること確認してください。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.stats.logFilesToKeep
現在のログファイルを含め、Integration Server がファイルシステムに保持する統計ログファイルの数を指定します。Integration Server が統計ログファイル数の上限に達すると、Integration Server が統計ログを循環するごとに、Integration Server はアーカイブ済みの最も古い統計ログファイルを削除します。watt.server.stats.logFilesToKeep を 1 に設定した場合、Integration Server は現在の統計ログファイルのみ保持し、以前の統計ログは保持しません。つまり、

Integration Server が統計ログを循環するとき、Integration Server は古い統計ログのアーカイブファイルを作成しません。set watt.server.stats.logFilesToKeep を 0 または 1 より小さい値に設定した場合、Integration Server は無制限の数の統計ログファイルを保持します。watt.server.stats.log.filesToKeep のデフォルト値は 0 です。

Integration Server が保持する統計ログファイルの数を減らしてから Integration Server を再起動した場合、Integration Server が統計ログを循環するまで、既存のログは削除されません。

Integration Server はアーカイブされる統計ログファイルに対して命名規則 *yyyymmdd_hhmmss*.csv (統計ログが 16 進数形式の場合は *yyymmdd_hhmmss*.log) を使用します。ここでタイムスタンプは Integration Server がログファイルをアーカイブした時刻を示します。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.stats.logRotateSize
Integration Server が統計ログファイルをロールオーバーするサイズを指定します。このプロパティは、N[KB|MB|GB] に設定します。N は、有効な任意の整数です。Integration Server が統計ログを循環する最小サイズは 33 KB です。単位として KB を使用する場合、N は 33 以上の値に設定する必要があります。単位を指定しない場合、Integration Server では N 値をバイトとして扱います。この場合、N は 32768 以上に設定しないと有効になりません。整数と測定単位の間にスペースを入れないでください。無効な値が指定されると、Integration Server はパラメータの値が指定されていないかのように処理を続行します。Integration Server Administrator の [**拡張設定**] ページを使用して値を設定すると、妥当性検査によって無効な値は保存されません。

このパラメータのデフォルト値はありません。デフォルトでは、パラメータの値はありません。watt.server.stats.logRotateSize の値が指定されない場合、Integration Server は watt.server.statsLogRotateInterval で指定された間隔でのみ統計ログを循環します。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.stats.packages.exclude
以下にサービスを含めないパッケージのカンマ区切りリストを指定します。

- 統計ログ
- [**サーバ**] > [**統計**] ページ
- [**サーバ**] > [サービスの使用状況] ページ
- **ダッシュボード** の [**使用状況**] タブ
- ダッシュボードの [**サービス**] タブにある [**キャッシュとプリフェッチの情報**] と [**サーキットブレーカー情報**] の表

  注:
  ダッシュボードの [**サービス**] タブにある [**サービスインスタンス**] タブには、除外サービスは表示されますが、[**最終実行日時**] と [**キャッシュ、プリフェッチ、サーキットブレーカー**] 列は空になります。

WmRoot のようにフルパッケージ名指定するか、または Wm* のように、パッケージ名の最初の文字の後にワイルドカード (*) を使用して指定します。このパラメータのデフォルト値は「Wm*」であり、名前が「Wm」で始まるすべてのパッケージが統計ログから除外されます。

このパラメータへの変更は、次に Integration Server が統計のためにポーリングしたときに反映されます。

watt.server.stats.pollTime
統計ログの更新間隔を秒数で指定します。スタンドアロン Integration Server では、watt.server.stats.pollTime は、0 (ゼロ) 以上の整数にする必要があります。クラスタの Integration Server では、watt.server.stats.pollTime は 0 (ゼロ) よりも大きく 60 未満の整数にする必要があります。デフォルトは 60 です。

Integration Server Administrator は、[**サーバ**] > [スケジューリング] > [システムタスクの表示] ページで、統計ログタスクの間隔としてこの値を表示します。

watt.server.statsLogRotateInterval
統計ログファイルのログ再利用間隔の長さ (分単位) を指定します。デフォルトは「`1440`」(24 時間) です。

Integration Server Administrator は、[**サーバ**] > [スケジューリング] > [システムタスクの表示] ページで、Stats ログ再利用タスクの間隔として watt.server.statsLogRotateInterval 値を表示します。Integration Server Administrator は、この値を分単位ではなく秒単位で表示します。たとえば、watt.server.logRotateInterval をデフォルト値の「1440」に設定した場合、Integration Server Administrator はログ再利用間隔の値を 86400 秒と表示します。システムタスクの表示の詳細については、769 ページの「スケジュールされているユーザタスクの表示」を参照してください。

Integration Server の起動後、Integration Server はすぐに watt.server.statsLogRotateInterval で指定された時刻に統計ログの循環を開始します。watt.server.statsLogRotateInterval を 2 分に設定し、Integration Server が午後 9:30 に起動された場合、Integration Server は 9:32、9:34、9:36 などに統計ログを循環します。この動作の例外として、ログ再利用時間が Integration Server の起動時刻から午前零時を超える場合、Integration Server は最初に午前零時に統計ログを循環します。その後、Integration Server は指定間隔でログを循環します。たとえば、watt.server.statsLogRotateInterval が 300 分 (5 時間) に設定され、Integration Server が午後 9:30 に起動された場合、Integration Server は午前零時に統計ログファイルを循環し、その後は 5 時間ごとに統計ログを循環します。

> **注:**
> watt.server.statsLogRotateInterval サーバ設定パラメータは、以前は watt.server.logRotateInterval という名前でした。

watt.server.storage.addKeyToStoreIfNotPresent
pub.storage:lock サービスの実行時、指定されたキーがデータストアに存在しない場合に、そのキーをデータストアに追加するかどうかを指定します。pub.storage:lock サービスの実行時、指定されたキーが存在しない場合に、そのキーをデータストアに追加するには、このパラメータを「true」に設定します。pub.storage:lock サービスは指定されたキーを作成し、それに NULL 値を割り当ててからデータストア内のエントリをロックします。pub.storage:lock サービスでキーをデータストアに追加しない場合は、このパラメータを「false」に設定します。指定されたキー値がデータストアに存在しない場合、pub.storage:lock サービスは NOP (動作なし) です。デフォルト値は「`false`」です。pub.storage サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.storage.lock.maxDuration
pub.storage サービスがロックを保持する最大時間 (ミリ秒単位) を指定します。デフォルト値は 180000 ミリ秒 (3 分) です。pub.storage サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.storage.lock.maxWait
pub.storage サービスがロックの獲得を待つ最大時間 (ミリ秒単位) を指定します。デフォルト値は 240000 ミリ秒 (4 分) です。pub.storage サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.storage.lock.sweepInterval
Integration Server で期限切れの pub.storage ロックを削除する頻度 (秒単位) を指定します。期限切れの pub.storage ロックとは、watt.server.storage.lock.maxDuration パラメータで指定した値が経過したロックです。デフォルトは 5 秒です。watt.server.storage.lock.sweepInterval パラメータを 1 未満の値に設定した場合、Integration Server はその設定を無視し、代わりに 5 秒を使用します。このプロパティの変更を適用するのに Integration Server を再起動する必要はありません。pub.storage サービスの詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

watt.server.storage.optimizeForStandalone
Integration Server が pub.storage 実装の最適化を使用するかどうかを指定します。「true」に設定した場合、Integration Server はデータベースキーをローカルメモリにキャッシュします。これによって、pub.storage のパフォーマンスが多少向上することがあります。「false」に設定した場合 (デフォルト)、Integration Server は pub.storage のデータベースキーをローカルメモリにキャッシュしません。

重要:
クラスタ内では、watt.server.storage.optimizeForStandalone を「true」に設定しないでください。pub.storage データベースにアクセスできなくなることがあります。

watt.server.storage.skipStoreExistCheck
pub.storage* サービスを実行するときに、Integration Server がデータストア名の妥当性を検査するかどうかを指定します。「true」に設定されている場合は、pub.storage* サービスを実行するときに、Integration Server がデータストア名の確認をスキップします。「false」に設定されている場合は、pub.storage* サービスを実行するときに、Integration Server がデータストア名を検証し、データストアが登録されているかどうかを確認します。Integration Server がデータストア名を認識しない場合は、Integration Server はデータストア名が無効であることを示すエラーをスローします。データストアを Integration Server に登録するには、pub.storage:registerStore サービスを使用します。デフォルト値は true です。

watt.server.strictAccessExceptionLogging
Integration Server が HTTP 401 Access Denied をエラーとしてログに記録して、通知を行うかどうかを指定します。このプロパティを「`true`」に設定すると、Integration Server は HTTP 401 Access Denied をエラーとしてログに記録し、通知を行います。「`false`」に設定すると、Integration Server は HTTP 401 Access Denied をエラーとしてログに記録せず、通知を行いません。デフォルトは「`false`」です。

watt.server.suppresscwarn
Designer で C サービスを実行するときに、Integration Server で変数の不足に関する警告メッセージをサーバログに書き込むかどうかを指定します。このプロパティを「`true`」に設定し

た場合、Integration Server はこれらのメッセージをサーバログに書き込みません。このプロパティを「false」に設定した場合、Integration Server は変数の不足に関するメッセージをサーバログに書き込みます。デフォルトは「false」です。

watt.server.sync.timeout
通知が発行された特定のキーに関するロックオブジェクトが存続する時間を指定します。pub.sync:notify サービスを呼び出して通知を作成すると、pub.sync:wait はその通知を受信できます。*key* に関する通知を待機しているスレッドは、待機要求の存続時間が通知の存続時間と重なっている限り、その通知を受信します。ただし、他のサービスが待機する前に、排他的待機が設定されたスレッドが通知の *key* に対して登録された場合、排他的待機で通知を受信すると同時にその通知は終了します。デフォルトは 60 秒です。

watt.server.systemtasks.debug
システムタスクに関連するデバッグログを有効にします。デフォルトで、値は「false」です。ログを有効にするには、値を「true」に変更します。ログは server.log のログ機能 [**ユーザタスクスケジューラ**] の下に書き込まれます。システムタスクログには、タスクの作成時、終了時、および定期的タスクの場合は再実行時などの情報が含まれます。

watt.server.thread.aging.limit
スレッドをスレッドプールに保持する時間の長さ (分) を指定します。この時間が経過し、スレッドが watt.server.thread.usage.limit パラメータで指定された最大回数を使用された場合、Integration Server はサービスが完了するまで待機してから、スレッドをプールから解放します。この値を -1 に設定すると、スレッドはスレッドプール内に無期限に保持されます。デフォルトは 60 分です。

注:
watt.server.thread.aging.limit と watt.server.thread.usage.limit の両方の値に達した場合に限り、Integration Server は期限切れのスレッドをリリースします。どちらのパラメータにも正の整数を設定する必要があります。どちらかのパラメータ、または両方のパラメータの値を -1 に設定した場合、どちらのパラメータも無効になります。

重要:
この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。

watt.server.thread.usage.limit
プールされたスレッドを使用できる最大回数を指定します。この最大回数に到達し、watt.server.thread.aging.limit パラメータで指定された時間が経過した場合、Integration Server はサービスが完了するまで待機してから、スレッドをプールから解放します。この値を -1 に設定すると、スレッドを無期限に再利用できます。デフォルトは 1000 です。

注:
watt.server.thread.aging.limit と watt.server.thread.usage.limit の両方の値に達した場合に限り、Integration Server は使用中のスレッドをリリースします。どちらのパラメータにも正の整数を設定する必要があります。どちらかのパラメータ、または両方のパラメータの値を -1 に設定した場合、どちらのパラメータも無効になります。

重要:
この設定はサーバのパフォーマンスとスループットに影響するため、特に注意して使用します。

watt.server.threadKill.enabled
スレッド強制終了機能を有効にするかどうかを制御します。この機能を使用すると、サービスが無応答状態になった場合にサービススレッドをキャンセルまたは強制終了できます。スレッドをキャンセルした場合、Integration Server はそのスレッドで保持されているリソースを解放します。また、スレッドを強制終了すると、Integration Server はそのスレッドを停止し、スレッドプールから解放し、スレッドプールを補充します。デフォルトは「true」です。スレッド強制終了機能を有効にすると、[**サーバ**] > [統計情報] > [システムスレッド] ページに移動してスレッドをキャンセルできます。スレッドをキャンセルまたは強制終了する方法の詳細については、754 ページの「サービスに関連付けられたスレッドのキャンセルと強制終了」を参照してください。

watt.server.threadKill.interruptThread.enabled
タイムアウトがトリガーされた、またはキャンセルされたサービスを Integration Server が中断するかどうかを指定します。「true」に設定した場合、タイムアウトがトリガーされたとき、またはサービスがキャンセルされたときに、Integration Server はサービスを実行しているサービススレッドを中断します。デフォルトは「`false`」です。

重要:
このプロパティの値を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.threadKill.timeout.enabled
Integration Server でフローステップのタイムアウト設定を処理する方法を制御します。フローステップには、ステップを実行できる期間を制御するタイムアウトプロパティがあります。デフォルト (true) では、ステップのタイムアウト期間が経過すると、Integration Server は FlowTimeoutException をスローし、フローの実行が次のステップに進みます。このプロパティを「false」に設定した場合、タイムアウト期間が経過してもフローステップを引き続き実行できることがあります。たとえば、フローステップが別のサービス (たとえば、ServiceA) を呼び出し、ServiceA の実行中にタイムアウト期間が経過した場合、その後も ServiceA は引き続き実行されます。ServiceA が終了すると、親フローサービスに制御が戻され、その親フローサービスが例外を発行します。

watt.server.threadPool
サーバが、サービスを実行するときに使用するスレッドプール内で管理する最大スレッド数を指定します。この最大スレッド数に達した場合、サーバは、現在のサービスが完了してスレッドがプールに戻されるまで待機してから、次のサービスを実行します。デフォルトは 75 です。

watt.server.threadPool.cloudRequests
Integration Cloud 要求の処理に使用できるサーバスレッドプールの最大パーセンテージを指定します。デフォルトは 5 です。

watt.server.threadPool.cloudRequests が「0」に設定されている場合、Integration Server は webMethods Cloud からの要求を一度に処理します。オンプレミスの Integration Server は、現在のサービスの実行が完了するまで、webMethods Cloud からの後続要求の処理をブロックします。これは、webMethods Cloud からの要求の処理が遅れる原因になります。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.threadPoolMin

サーバが、サービスを実行するときに使用するスレッドプール内で管理する最小スレッド数を指定します。サーバ起動時のスレッドプールには、初期値としてここで指定された最小スレッド数が設定されます。watt.server.threadPool 設定で指定した最大スレッド数に達するまで、サーバは必要に応じてスレッドをプールに追加します。デフォルトは 10 です。

watt.server.transaction.ignore.exception
Integration Server が、明示的または暗示的トランザクションがコミットされるときに発生した例外を無視するかどうかを指定します。「false」に設定されている場合、明示的または暗示的トランザクションのコミット中 (pub.art.transaction.commitTransaction サービスの実行中など) にエラーが発生すると、Integration Server は例外を反映してサービスの実行を停止します。「true」に設定されている場合、Integration Server は例外を無視して、フローサービスの次のステップを実行します。デフォルト値は「false」です。

明示的または暗示的トランザクションをコミットしている間に発生した例外を反映しサービスを停止するのは、正しい動作です。しかし、状況によっては、トランザクションサービスに対する例外の適切な処理が望ましくない場合があります。例えば、フローサービスが別のローカルトランザクションを含んでいる場合、pub.art.transaction.commitTransaction サービスに対して例外を反映した結果、2 つのローカルトランザクションを持つ 1 つのフローサービスとなることがあります。1 つのトランザクションコンテキストは任意の数の XA_TRANSACTION 接続を含むことができますが、1 つより多くの LOCAL_TRANSACTION を含むことはできません。このような場合は、例外を無視するという正しくない動作のほうがより好ましい動作となります。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.transaction.recovery.abandonTimeout
Integration Server が未完了の XA トランザクションを解決しようとしているときにエラーが発生した場合に、Integration Server で解決を試行する最長時間 (分単位) を指定します。デフォルトは 5 分です。

watt.server.transaction.recovery.sleepInterval
Integration Server が未完了の XA トランザクションを解決しようとしているときにエラーが発生した場合に、Integration Server で次に解決を試行するまで待機する時間 (秒単位) を指定します。デフォルトは 30 秒です。

watt.server.transaction.xastore.maxTxnPerFile
XA 回復ログファイル内の一意の XA トランザクションの最大数を指定します。XA 回復ログファイルがトランザクションの最大数に達すると、Integration Server は新しいファイルを作成します。デフォルトは 2000 トランザクションです。

アクティブな XA トランザクションが 2000 を超え、入出力のために Integration Server のパフォーマンスが遅延している場合、XA 回復ログファイルの一意の XA トランザクションの最大数を増やすことを検討します。各ファイルで許可される一意の XA トランザクションの数を増やすと、XA 回復ログに使用されるファイルの数が減ります。その結果、Integration Server が XA 回復の実行時に検索するファイル数が少なくなる可能性があります。

ファイルに保存される一意の XA トランザクション数を減らすと、開いているトランザクションを読み取り、統合する時間が減り、トランザクション回復時に役立つことがあります。

watt.server.transaction.xastore.performXALogging
Integration Server が XA 回復ストアにトランザクション情報を書き込むかどうかを指定します。true に設定すると、Integration Server は各 XA トランザクションの状態と進捗に関する

情報をログに記録します。false に設定すると、Integration Server は XA トランザクション情報のロギングをスキップします。デフォルトは「true」です。

重要:
このプロパティを false に設定した場合、Integration Server はトランザクションの処理中に XA 回復ストアに情報を記録しないため、トランザクションの回復は不可能になります。Integration Server が未完了トランザクションを自動的に解決するようにするか、未完了トランザクションを手動で解決する場合、Integration Server は XA ロギングを実行する必要があります。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.trigger.interruptRetryOnShutdown
Integration Server をシャットダウンする要求が、webMethods messaging trigger サービスの再試行プロセスを中断するかどうかを指定します。このパラメータが「`false`」に設定されていると、Integration Server は最大再試行回数に達するまで待機してからシャットダウンします。また、Integration Server は、再試行処理中にトリガーサービスが正常に実行された場合もシャットダウンします。このパラメータが「`true`」に設定されていると、Integration Server は現在のサービスの再試行が完了するのを待機します。webMethods messaging trigger サービスをもう一度再試行する必要がある (ISRuntimeException が原因でサービスが終了した) 場合でも、Integration Server は再試行プロセスを停止し、シャットダウンします。再起動時、転送手段 (Broker、Universal Messaging、またはローカルパブリッシュの場合は一次ストア) によって、ドキュメントが処理のためにトリガーに再デリバーされます。デフォルトは「`false`」です。

watt.server.trigger.keepAsBrokerEvent
Broker からのドキュメントの取得時に通常は webMethods messaging trigger に代わって Integration Server で実行されるデコードを省略するかどうかを指定します。このプロパティを「`true`」に設定した場合、Integration Server は $brokerEvent という名前のオブジェクトを使用してトリガーサービスに Broker イベントの値を渡すだけで、デコードは行いません。Integration Server が Broker のネイティブイベントを受信する場合は、このパラメータを「`true`」に設定します。デフォルトは「`false`」です。

Broker ネイティブイベントのパブリッシュの詳細については、『*Publish-Subscribe Developer's Guide*』を参照してください。

注:
webMethods Broker が廃止されたため、このパラメータは廃止されました。

watt.server.trigger.local.checkTTL
ローカルにパブリッシュされたドキュメントの廃棄までの時間を、Integration Server が厳密に適用するかどうかを指定します。このパラメータを「`true`」に設定した場合、トリガーキューの中のローカルにパブリッシュされたドキュメントの処理前に、Integration Server はドキュメントの期限が切れていないかどうかを確認します。期限切れのドキュメントは、Integration Server で廃棄されます。デフォルトは「`false`」です。

watt.server.trigger.managementUI.excludeList
Integration Server Administrator の [**webMethods トリガー**] ページから除外するトリガーをカンマ区切りのリストで指定します。また、Integration Server では、すべてのトリガーのドキュメント抽出やドキュメント処理を一時停止または再開するトリガー管理の変更を行う場

合でも、これらの webMethods messaging triggerは対象になりません。ただし、Integration Server でグローバルトリガー制御 ([キュー容量のスロットル] や [実行スレッドのスロットル]) を使用して容量、補充レベルまたは実行スレッドの最大数を変更する場合は、これらの webMethods messaging triggerも対象になります。

除外するすべての webMethods messaging triggerの完全修飾名を指定することができます。また、完全修飾名の冒頭部分にアスタリスク (*) を付けたものを指定して、パターンマッチングを使用してトリガーのグループを除外することも可能です。Integration Server は指定されたパターンで始まるトリガーをすべて除外します。たとえば、pub.prt フォルダにあるトリガーをすべて除外する場合は、次のように指定します。

```
watt.server.trigger.managementUI.excludeList = pub.prt*
```

Integration Server が webMethods messaging triggerに関するリソース監視サービスを実行する間隔を秒単位で指定します。リソース監視サービスは、トリガーサービスによって使用されるリソースの使用可能性をチェックするためにユーザが作成するサービスです。再試行の試みがすべて失敗したために webMethods messaging triggerが一時停止された場合、Integration Server はリソース監視サービスを実行して、すべてのリソースが使用可能であるかどうかを確認します。デフォルトは 60 秒です。

リソース監視サービスの詳細については、『*Publish-Subscribe Developer's Guide*』を参照してください。

watt.server.trigger.preprocess.monitorDatabaseOnConnectionException
トリガー処理中に ConnectionException で一時的エラーが発生した場合に、Integration Server がシステムタスクをスケジュールして、ドキュメント履歴データベースを監視するかどうかを示します。ConnectionException は、ドキュメント履歴データベースが重複抑制処理に対して有効になっていないまたは適切に設定されていないことを示します。

このプロパティを「true」に設定すると、Integration Server はドキュメント履歴データベースへの接続を監視する内部サービスを実行するシステムタスクをスケジュールします。ドキュメント履歴データベースで重複抑制処理が有効になっているまたは適切に設定されると、内部サービスは、ドキュメント履歴データベースへの接続を使用できることを示します。次に、ドキュメント履歴データベースへの接続が使用可能であることが判明すると、Integration Server はトリガーを再開して再実行します。

このプロパティを「false」に設定すると、Integration Server はデータベースの使用可能性をチェックするシステムタスクをスケジュールせず、トリガーを自動的に再開しません。ドキュメント履歴データベースを正しく再開した後、トリガーを手動で設定する必要があります。

デフォルトは「false」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.trigger.preprocess.suspendAndRetryOnError
トリガー実行のプリプロセスフェーズでエラーが発生した場合に、Integration Server がトリガーを一時停止するかどうかを指定します。プリプロセスフェーズとは、ローカルキューからトリガーがドキュメントを取得してからトリガーサービスが実行されるまでの期間を意味します。このプロパティを「`true`」に設定すると、トリガープロパティ **On Retry Failure** が **Suspend に設定されて再試行されるか**、トリガープロパティ **On Transaction Rollback** が **Suspend に設定されて回復し**、Integration Server は処理中に次のうちの 1 つが発生した場合にトリガーは一時停止します。

- Integration Server がトリガーの重複検出を実行したときにドキュメント履歴データベースが使用不可であったために、一時的エラーが発生します。トリガー処理中の一時的なエラーの処理の詳細については、*webMethods Service Development Help*を参照してください。

  ドキュメント履歴データベースが正しく設定されている場合、Integration Server はトリガーを一時停止し、ドキュメント履歴データベースの使用可能性をチェックするサービスを実行するシステムタスクをスケジュールします。このサービスによりドキュメント履歴データベースが使用可能であることが判明すると、Integration Server はトリガーを再開して再実行します。

  ドキュメント履歴データベースが適切に設定されていない場合は、Integration Server は ConnectionException をスローします。トリガープロセス中に Integration Server が ConnectionException を処理する方法を決定するには、watt.server.trigger.preprocess.monitorDatabaseOnConnectionException を参照してください。

- ISRuntimeException によりドキュメントリゾルバサービスが終了した場合。Integration Server は、トリガーを一時停止し、トリガーのリソース監視サービスが指定されていれば、このサービスを実行するためのシステムタスクをスケジュールします。リソース監視サービスにより、トリガーで使用されるリソースが使用可能であることが判明すると、Integration Server はトリガーを再開し、再度トリガーの実行を試みます。リソース監視サービスが指定されていない場合は、ユーザが手動でトリガーを再開する必要があります (Integration Server Administrator を使用するか、pub.trigger* サービスを使用します)。

このプロパティが「`false`」に設定され、トリガープロパティ **On Retry Failure** が「**Throw Exception**」に設定されるか、トリガープロパティ **On Transaction Rollback** が「**Recover only**」に設定されている場合は、Integration Server はトリガプリプロセスエラーが発生した場合にトリガーを一時停止しません。ドキュメント履歴データベースが使用できない場合、Integration Server は以下の処理を行います。

- トリガーでドキュメントリゾルバサービスが指定されている場合、Integration Server はドキュメントリゾルバサービスを実行して、ドキュメントの状態を確認します。ISRuntimeException によりドキュメントリゾルバサービスが終了した場合、Integration Server はドキュメントに「インダウト」という状態を割り当て、ドキュメントの受信確認を行い、監査サブシステムを使用してドキュメントを記録します。

- ドキュメントリゾルバサービスが指定されていない場合、Integration Server はドキュメントに「インダウト」という状態を割り当て、ドキュメントの受信確認を行い、監査サブシステムを使用してドキュメントを記録します。

注:
Integration Server は監査システムを使用して webMethods messaging trigger のドキュメントをログに記録します。

watt.server.trigger.preprocess.suspendAndRetryOnError のデフォルト値は「`false`」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

リソース監視サービスの構築の詳細については、『*Publish-Subscribe Developer's Guide*』または『*Using webMethods Integration Server to Build a Client for JMS*』を参照してください。

トリガー処理中の一時的なエラーの処理の詳細については、*webMethods Service Development Help*を参照してください。

watt.server.trigger.removeSubscriptionOnReloadOrReinstall
トリガーを含むパッケージが再ロードされた場合、または更新パッケージがインストールされた場合に、Integration Server が webMethods messaging triggerのドキュメントタイプサブスクリプションを削除するかどうかを指定します。このプロパティを「`true`」 (デフォルト)に設定した場合、パッケージの再ロードまたは更新パッケージのインストールが行われると、Integration Server は、パッケージ内の webMethods messaging triggerのドキュメントタイプサブスクリプションをすべて削除してから再作成します(Integration Server が Broker に接続している場合、Integration Server は、Broker 上のトリガークライアントに対してサブスクリプションの削除と再作成を行います)。これにより、ドキュメントタイプサブスクリプションが存在しない時間が短時間発生します。この間、トリガーは通常であればサブスクライブするドキュメントを受信できません。

このプロパティを「`false`」に設定した場合、パッケージの再ロードまたは更新が行われても、Integration Server は webMethods messaging triggerのドキュメントタイプサブスクリプションの削除と再作成を行いません。Integration Server はトリガーの新しいドキュメントタイプサブスクリプションを作成します。ただし、Integration Server は既存のサブスクリプションの変更は行いません。つまり、トリガーでドキュメントタイプサブスクリプションが削除されていた場合でも、パッケージの再ロードまたは更新時に、このサブスクリプションは削除されません。このため、このプロパティを「`false`」に設定した場合、Broker 上のトリガークライアントに削除済みのドキュメントタイプサブスクリプションが残っているために、トリガーで既にサブスクライブを停止しているドキュメントタイプが受信されることがあります。webMethods Broker バージョン 6.5.2 を使用する場合は、My webMethods を使用して、Broker 上のトリガークライアントから、不要なドキュメントタイプサブスクリプションを削除できます。デフォルトは「`true`」です。

注:
このプロパティは Integration Server のクラスタを実行している webMethods messaging trigger に影響しません。

watt.server.trigger.reuseSession
ドキュメントロケールが Integration Server のデフォルトロケールと同じ場合に、webMethods messaging triggerの複数のインスタンスで、Integration Server 上の同じセッションが使用されるかどうかを指定します。このプロパティを「`true`」に設定すると、Integration Server はドキュメントの処理前にドキュメントのロケールをチェックします。ドキュメントのロケールが Integration Server のデフォルトロケールと同じ場合、またはロケールが指定されていない場合、トリガーは共有セッションを使用します。ドキュメントのロケールがデフォルトロケールと異なる場合、Integration Server は、トリガーがそのドキュメントの処理に使用する新しいセッションを作成します。このプロパティを「`false`」に設定した場合、Integration Server はトリガーのインスタンスごとに新しいセッションを使用します。デフォルトは「`false`」です。

webMethods messaging trigger でセッションを再使用することにより、パフォーマンスが向上する可能性があります。ただし、このプロパティはすべてのアダプタに適用されるわけではありません。

注:
Adapter for SAP を使用する場合、同時トリガーに pub.sap.client:lockSession および pub.sap.client:releaseSession サービスを呼び出すマルチステップ SAP トランザクション

を実行するトリガーサービスがある場合は、watt.server.trigger.reuseSession を「false」に設定します。これらのサービスには専用セッションが必要です。

watt.server.trigger.shutdown.timeout
Broker からメッセージを取得した同時実行 webMethods messaging trigger を無効にした後に、Integration Server がトリガーサービスの実行の終了を待つ時間を秒単位で指定します。webMethods メッセージングトリガーは Integration Server Administrator または組み込みサービスを使用して明示的に無効にするか、トリガーが含まれているパッケージを停止 (再ロードまたは無効化) することにより暗黙的に無効にできます。トリガーが無効になると、トリガーによって呼び出された任意のトリガーサービスが実行中の場合、トリガーを正常にをシャットダウンする前に、Integration Server は watt.server.trigger.shutdown.timeout に指定された時間の長さだけ待機します。トリガーのシャットダウンが発生した後、完了するために実行されているトリガーサービスに対して、Integration Server はメッセージを確認できません。トリガーは、トリガーの起動時にメッセージを再取得する可能性があります。デフォルトは 3 秒です。Integration Server は、Broker からメッセージを取得する webMethods messaging trigger が次に無効になるときに、新しい値を使用します。

watt.server.trigger.suspendOnAuditErrorWhen
Integration Server で webMethods messaging triggerを一時停止するかどうかを指定します。次の両方に該当する場合、トリガーを一時停止できます。

- トリガーサービスが失敗する。
- 監査例外の発生が原因で、トリガーサービスが監査データを監査データベースに書き込めない。

Integration Server が webMethods messaging triggerを一時停止すると、そのトリガーのドキュメント処理およびドキュメント抽出が停止されます。トリガーを一時停止すると、Integration Server でのドキュメントの受信確認、およびメッセージングプロバイダでのドキュメントの廃棄が行われなくなります。トリガーを一時停止した後、Integration Server は、監査データベースへの接続を監視し、その接続が再確立されたときにトリガー (ドキュメント処理およびドキュメント抽出) を再開します。Integration Server は、受信確認の済んでいないドキュメントをメッセージングプロバイダから抽出し、再び処理しようとします。

watt.server.trigger.suspendOnAuditErrorWhen パラメータは、次のいずれかの値に設定します。

指定内容	[宛先]
`Never`	エラーが原因でトリガーサービスが終了し、トリガーサービスがデータを監査データベースに書き込むときに監査例外が発生する場合に、Integration Server でそのトリガーを一時停止しないように指定します。
`[エラー]`	エラーが原因でトリガーサービスが終了し、トリガーサービスがデータを監査データベースに書き込むときに監査例外が発生する場合に、Integration Server でそのトリガーを一時停止するように指定します。
`ErrorPipelineEnabled`	エラーが原因でトリガーサービスが終了し、監査ログにサービスパイプラインを含めるようにトリガーサービスが設定されており、トリガーサービスがデータを監査データベースに書き込むと

指定内容	[宛先]
	きに監査例外が発生する場合に、Integration Server でそのトリガーを一時停止するように指定します。デフォルトは「ErrorPipelineEnabled」です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.server.trigger.suspendOnAuditErrorWhen 設定パラメータは、データを同期的に書き込むように監査サブシステムが設定されている場合にのみ適用されます。

watt.server.tspace.location

Integration Server で、メモリに保持する代わりに、サイズの大きいドキュメントを一時的に格納するハードディスクドライブ領域の絶対ディレクトリパスを指定します。Integration Server でこのディレクトリに格納される各ファイルには、DocRes*xxxxx*.dat という名前が付けられます。この *xxxxx* は、任意の長さおよび文字列の値です。絶対ディレクトリパスを、Integration Server がインストールされているコンピュータのディレクトリに指定します。デフォルト値は JVM の一時ディレクトリ (java.io.tmpdir の値など) です。

例: Integration Server が D ドライブの LargeDocTemp ディレクトリを使用するようにするには、次のように指定します。

```
watt.server.tspace.location=D:\LargeDocTemp
```

Integration Server のクラスタを使用している場合は、各サーバに独自の Tspace が必要になります。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.server.tspace.max

`watt.server.tspace.location` プロパティを使用して定義したハードディスクドライブ領域に、一度に格納可能な最大バイト数を指定します。指定したバイト数を超える大きいサイズのドキュメントを Integration Server がハードディスクドライブ領域に書き込もうとすると、エラーメッセージがサーバのコンソールに表示されます。また、このドキュメントは格納されません。正の整数のバイト数を指定します。デフォルト値は 52,428,800 バイト (50 MB) です。

例: 30,000,000 バイトまで格納可能な最大バイト数を設定するには、次のように指定します。

```
watt.server.tspace.max=30000000
```

ヒント:
ドキュメントを一時的に保存するためのハードディスクドライブ領域のサイズは、同時に処理するドキュメントの数と、処理するドキュメントのサイズによって異なります。たとえば、通常の同時処理ドキュメント数が 10 である場合は、同時処理ドキュメントの合計サイズの 10～15 倍のハードディスクドライブ領域が必要です。

重要:
このプロパティの値を変更した後、Integration Server を再起動する必要があります。

watt.server.tspace.timeToLive

ドキュメントが削除されるまでTspace内にドキュメントが保持される時間をミリ秒単位で指定します。このパラメータで定義された値より長い時間ドキュメントがTspace内に保持されると、ドキュメントの有効期限が切れ、Tspaceからの削除の対象となります。Tspaceの空きディスクスペースが不足し始めると、期限切れのTspaceドキュメントは削除されます。デフォルトは0です。

重要:
このプロパティの値を変更した後、Integration Serverを再起動する必要があります。

watt.server.tx.cluster.jobPendingWait
Integration Serverが、TContext.invokeTx()またはTContext.retrieveTx()のジョブの結果が取得されたときに、保証付きデリバージョブが完了するまで待つための時間を秒単位で指定します。デフォルトは60秒です。

重要:
パラメータの変更を反映するには、796ページの「保証付きデリバーの再初期化」に記述されているように保証付きデリバーを再初期化するか、Integration Serverを再起動する必要があります。

watt.server.tx.cluster.lockBreakSecs
クラスタジョブストア上のジョブのロックを解除するまでクラスタサーバが待機する秒数を指定します。デフォルトは120です。

watt.server.tx.cluster.lockTimeoutMillis
クラスタジョブストア内のジョブに対して設定された更新ロックの間隔でクラスタサーバがスリープする時間をミリ秒単位で指定します。デフォルトは100です。

watt.server.tx.heuristicFailRetry
障害発生後にIntegration Serverが再起動したときに、ペンディングになっているジョブストア内の保証付きデリバートランザクションに対してIntegration Serverでサービスを再実行するかどうかを指定します。トランザクションがペンディングの場合は、Integration Serverで障害が発生する前にサービスの実行が開始されています。

設定を「`true`」にすると、Integration Serverはトランザクション状態を「**ペンディング (Pending)**」から「**新規 (New)**」にリセットし、サービスは再実行されます。設定を「`false`」にすると、Integration Serverはトランザクション状態を「**ペンディング (Pending)**」からヒューリスティックな障害を示す「**失敗 (Fail)**」にリセットし、サービスは再実行されません。デフォルトは「`true`」です。

watt.server.tx.sweepTime
受信する保証付きデリバートランザクションのジョブストアをスイープする (消去する) 間隔を秒数で指定します。サーバは、期限切れトランザクションを削除するためにジョブストアをスイープします。デフォルトは60です。

watt.server.txMail
保証付きデリバー機能がエラーのため使用できない場合 (Integration Serverにディスクフルが発生した場合や監査トレールログが満杯になった場合など) に通知する管理者の電子メールアドレスを指定します。デフォルトはありません。

watt.server.typeSynchronizer.verbose
ドキュメントタイプがBrokerと同期するときに冗長ログを有効にするかどうかを指定します。冗長ログはドキュメントタイプの同期をデバックするときに便利です。「true」に設定す

ると、ドキュメントタイプが Broker と同期するときに冗長ログを有効にします。Integration Server は wrapper.log にログメッセージを書き込みます。「false」に設定すると、ドキュメントタイプが Broker と同期するときに冗長ログを無効にします。デフォルトは「false」です。

watt.server.um.producer.transaction.commitRetryCount
トランザクションの障害のためにパブリッシュの最初の試行が失敗した後、Integration Server が Universal Messaging サーバに保証付きドキュメントをパブリッシュする試行数を指定します。再試行の最大数は 9 です。9 より大きい値を割り当てようとした場合、Integration Server ではプロパティの値が自動的に 9 に設定されます。デフォルトは 0 です。

再試行の値を設定するときは、トランザクションタイムアウトの合計値が MaxTransactionTime プロパティの値を超えないようにします。トランザクションタイムアウト合計値は、再試行の合計数を EventTimeout プロパティの値で乗算して計算します。たとえば、再試行の値を 9 に設定し、EventTimeout を 60s に設定した場合、トランザクションタイムアウト合計は 60(9+1) = 600s です。

注:
MaxTransactionTime および EventTimeout プロパティの値は Universal Messaging Enterprise Manager で設定できます。

watt.server.url.alias.partialMatching
Integration Server で URL エイリアスの部分一致を有効にするかどうかを指定します。このサーバ設定パラメータを「`true`」に設定し、Integration Server Administrator で URL エイリアスを定義した場合、Integration Server は URL エイリアスの部分一致を有効にします。デフォルトは「`false`」です。

部分一致が有効化されると、Integration Serverは、エイリアス全体が要求 URL の最初の文字からのすべてまたは一部と一致する場合、エイリアスを一致と見なします。

部分一致と URL エイリアスの定義の詳細については、449 ページの「HTTP URL エイリアスの設定」を参照してください。従来の方法を使用して設定された REST リソースの部分一致の詳細については、『*REST Developer's Guide*』を参照してください。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.server.userFtpRootDir
Integration Server が起動時に作成する FTP ルートディレクトリを指定します。Integration Server ユーザが FTP リスナーにログインすると、サーバはそのユーザの FTP ホームディレクトリを FtpRoot/*username* などの FTP ルートディレクトリに作成します。このルートディレクトリには、マッピングされたネットワークディレクトリを含め、任意のディレクトリを指定することができます。 このプロパティが定義されていない場合は、userFtpRoot という名前のデフォルトのディレクトリが Integration Server のホームディレクトリに作成されます。

FTP リスナーを介して Integration Server に接続しているユーザは、デフォルトの FTP ルートディレクトリ、または watt.server.login.userFtpDir プロパティで定義されたクライアントユーザディレクトリのいずれかに配置されます。

Administrator、Replicator および非特権ユーザは、以下のディレクトリで Put 操作および Get 操作を行うことができます。

ユーザ	アクセス可能なディレクトリ
Administrator	■ admin (Integration Server のホームディレクトリ内) ■ Administrator の固有のユーザディレクトリ ■ WmRoot を含め、すべてのパッケージのネームスペース全体 ■ 他のすべてのユーザディレクトリ
Replicator	■ *Integration Server_directory* \instances*instance_name*\replicate ディレクトリ ■ Replicator ACL 以下の ACL を持つ、ネームスペース内の任意のサービス ■ Replicator の固有のユーザディレクトリ
非特権ユーザ	■ ユーザ固有のユーザディレクトリ ■ ユーザの ACL 以下のレベルの ACL を持つ任意のサービス

ユーザが、固有のユーザディレクトリで Put コマンドを完了すると (サーバ側で STOR コマンドが完了したが、サーバがリターンコード 226 でクライアントに応答する前)、イベントが起動され、pub.client.ftp:putCompletedNotification ドキュメントを webMethods Broker にパブリッシュすることによって関係者に通知します。EDI パッケージはこのドキュメントをサブスクライブし、サーバに置かれたこのファイルを取り出します。

注:
STOU コマンドは、Integration Server ではサポートされていません。ただし、クライアントではサポートされています。『*webMethods Integration Server Built-In Services Reference*』で、組み込みサービス pub.client.ftp、pub.client.ftp:put および pub.client.ftp:mput を参照してください。

watt.server.ws.71xHandlerChainBehavior
7.1x で作成された Web サービス記述子のハンドラを Integration Server で実行するときに、7.1x のハンドラチェーン処理の動作を使用するかどうかを指定します。「true」に設定した場合、Integration Server は、7.1x の Web サービス記述子に対して Integration Server バージョン 7.1x で使用可能なハンドラチェーン処理の動作を使用します。「false」に設定した場合、Integration Server は、7.1x の Web サービス記述子に対して Integration Server 8.0 で使用可能なハンドラチェーン処理の動作を使用します。デフォルトは「`false`」です。

watt.server.ws.defaultNamespace
これは内部プロパティです。変更しないでください。

watt.server.ws.defaultPrefix
SOAP 障害の詳細で使用されるネームスペース http://www.webMethods.com/2001/10/soap/encoding に割り当てられるプリフィックスを指定します。デフォルトは「webM」です。

watt.server.ws.responseTNS.from.request
[**8.2 より前のバージョンとの互換モード**] プロパティが「true」に設定されているプロバイダ Web サービス記述子では、応答のトップレベルの要素のターゲットネームスペースに要求の

トップレベルの要素のターゲットネームスペースを使用するかどうかを指定します。「true」に設定すると、Integration Server によって送信される Web サービス応答は、要求のトップレベルの要素のターゲットネームスペースを応答のトップレベルのネームスペースとして使用します。これは正しくない動作ですが、下位互換性を提供します。クライアントが応答のトップレベルの要素のネームスペースを使用した応答を想定したために既存の Web サービスクライアントが失敗しても、クライアントを変更せずに既存のクライアントに応答の処理を続行させる場合は、watt.server.ws.responseTNS.from.request を「true」に設定します。

「false」に設定すると、Integration Server はプロバイダ Web サービス記述子のターゲットネームスペースを応答のトップレベルの要素のネームスペースとして使用します。これが正しい動作です。デフォルト値は「false」です。

注:
9.10 よりも前のバージョンの Integration Server から Integration Server 9.10 以降にアップグレードし、Web サービスクライアントで **8.2 より前のバージョンとの互換モード** = true に指定し、既存の Web サービスクライアントが応答のネームスペースと要求のネームスペースが一致することを想定する場合は、ネームスペースのミスマッチにより失敗します。Software AG は、現在のプロバイダ Web サービス記述子から取得した WSDL ドキュメントを使用してこれらの Web クライアントサービスを再生成することをお勧めします。また、watt.server.ws.responseTNS.from.request を「true」に設定すると、応答で現在の WSDL ドキュメントで指定されたネームスペースを使用するのではなく、要求のネームスペースを使用する動作に戻します。この動作は下位互換性を可能にしますが、プロバイダ Web サービス記述子の現在の WSDL に一致しない応答を生成することがあります。このため、Software AG は、watt.server.ws.responseTNS.from.request を「true」に設定することをお勧めしていません。

注:
Integration Server 7.1 で導入された Web サービス実装が Integration Server 10.4 以降は廃止されたため、このパラメータは廃止されました。Integration Server バージョン 7.1 で導入された Web サービス実装は 8.2 より前の互換モードで実行される Web サービス記述子を処理します。

watt.server.ws.security.timestampMaximumSkew
タイムスタンプの期限切れの妥当性検査が失敗しないように、Web サービスクライアントおよびホストのクロック間の最大許容誤差を秒数で指定します。Integration Server が受信 SOAP メッセージの妥当性検査を行うのは、メッセージの作成タイムスタンプが、タイムスタンプの最大スキュー値と現在のシステムクロック時間の合計に満たない場合だけです。

Integration Server で WS セキュリティポリシーを使用してセキュリティを実装しているときに、[Web サービスエンドポイントエイリアスの作成] ページで [**タイムスタンプの最大スキュー**] フィールドの値を設定していない場合、このパラメータを使用してタイムスタンプの最大スキュー値を設定できます。値は正の整数または 0 にする必要があります。デフォルトは 300 秒です。

watt.server.ws.security.timestampPrecisionInMilliseconds
送信メッセージのセキュリティヘッダーの Timestamp エレメントに表示されるタイムスタンプを秒精度にするか、ミリ秒精度にするかを指定します。ミリ秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss:SSS'Z'` を使用します。秒精度を設定した場合、Integration Server はタイムスタンプ形式として `yyyy-MM-dd'T'HH:mm:ss'Z'` を使用します。

Integration Server で WS ポリシーを使用して WS セキュリティを実装しているときに、[Web サービスエンドポイントエイリアスの作成] ページで [**タイムスタンプの精度**] フィールドの値を設定していない場合、このパラメータを使用して精度を設定できます。タイムスタンプの精度をミリ秒に設定する場合は、このプロパティを「`true`」に設定します。タイムスタンプの精度を秒に設定する場合は、このプロパティを「`false`」に設定します。デフォルトは「`true`」です。

watt.server.ws.security.timestampTimeToLive
送信メッセージの廃棄までの時間を秒単位で指定します。Integration Server は、この値を使用して、送信メッセージの Timestamp エレメントに期限を設定します。

Integration Server で WS ポリシーを使用して WS セキュリティを実装しているときに、[Web サービスエンドポイントエイリアスの作成] ページで [**タイムスタンプの破棄までの時間 (TTL)**] フィールドの値を設定していない場合、このパラメータを使用して廃棄までの時間の値を設定できます。値は 0 より大きい整数にする必要があります。デフォルトは 300 秒です。

watt.server.ws.security.usernameTokenTTL
UsernameToken の作成時刻 (wsu:Created エレメントで指定) とサーバへの到達時刻との間の許容される時間差 (秒単位) を指定します。このパラメータを使用して、ユーザ名トークン TTL をグローバルレベルで設定できます。デフォルトは 300 秒です。

また、[**設定**] > [**Web サービス**] ページの [**WS セキュリティプロパティ**] セクションを使用して、個々のエンドポイントエイリアスに対してこのプロパティを設定することもできます。[**設定**] > [**Web サービス**] ページで設定された値はこの設定の値を上書きします。

watt.server.ws.security.usernameTokenFutureTTL
未来のタイムスタンプを持つ wsu:Created エレメントと Integration Server のクロック間の許容される時間差を指定します。要求の作成時刻が、この設定で指定された値 (秒単位) 分だけ Integration Server への到達時刻を超えない場合、Integration Server はそのような要求を有効と見なします。デフォルトは 60 秒です。

また、[**設定**] > [**Web サービス**] ページの [**WS セキュリティプロパティ**] セクションを使用して、個々のエンドポイントエイリアスに対してこのプロパティを設定することもできます。[**設定**] > [**Web サービス**] ページで設定された値はこの設定の値を上書きします。

watt.server.wsdl.debug
Integration Server での WSDL の生成中または消費中に、デバッグ情報を標準出力に出力し、スタックトレースを標準エラーに出力するかどうかを指定します。「`true`」に設定した場合、Integration Server はデバッグ情報を標準出力に出力し、スタックトレースを標準エラーに出力します。デフォルトは「`false`」です。

watt.server.xml.encoding
受信 XML ファイルを処理するときに Integration Server で使用する必要があるエンコーディングを指定します。

XML ヘッダーでエンコーディングが指定されていない場合、Integration Server は http または ftp 要求の文字セットエンコーディングを使用して XML ファイルの処理を試みます。要求ヘッダーで文字セットエンコーディングを使用できない場合、Integration Server は watt.server.xml.encoding サーバ設定パラメータで指定された文字エンコーディングを使用します。このパラメータのデフォルト値はありません。

注:

watt.server.fileEncoding パラメータで指定した文字エンコーディングを使用して受信 XML ファイルを処理するように Integration Server を設定した場合は、watt.server.fileEncoding パラメータの値が watt.server.xml.encoding に指定した値と同じものに設定されていることを確認してください。XML ファイルを処理するように watt.server.fileEncoding を設定していない場合は、この修正をインストールした後、または Integration Server のより新しいバージョンにアップグレードした後、受信 XML ファイルを処理するように watt.server.xml.encoding を設定できます。watt.server.fileEncoding を使用して、受信 XML ファイル以外のすべてのファイルを処理できます。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

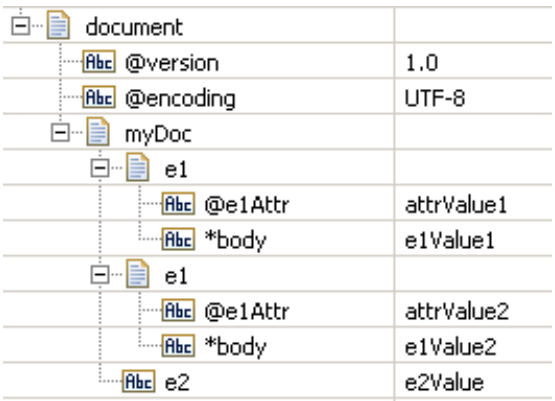
watt.server.xml.enforceEntityRef
XML パーサが不正なエンティティを検出したときにサーバが例外をスローするかどうかを指定します。この値が「`true`」に設定されていると、XML または DTD で不正なエンティティが検出されたときにサーバが例外をスローします。この値が「`false`」(デフォルト) に設定されていると、サーバは不正なエンティティを許容し、例外をスローしません。

watt.server.xml.xmlNodeToDocument.keepDuplicates
配列が作成されない場合 (*makeArrays* 入力パラメータが「false」に設定されている場合) に、pub.xml:xmlNodeToDocument サービスで XML ノードの複数箇所に出現するエレメントを保持するかどうかを指定します。このパラメータでは、サービスに渡される XML ドキュメントに出現するエレメントを pub.event.eda:eventToDocument サービスで保持するかどうかも指定します。「true」に設定した場合、pub.xml:xmlNodeToDocument サービスによって生成されるドキュメントでは、複数箇所に出現するエレメントが保持されます。「false」に設定した場合、pub.xml:xmlNodeToDocument サービスによって生成されるドキュメントでは、最後に出現するエレメントだけが保持されます。デフォルトは「`true`」です。

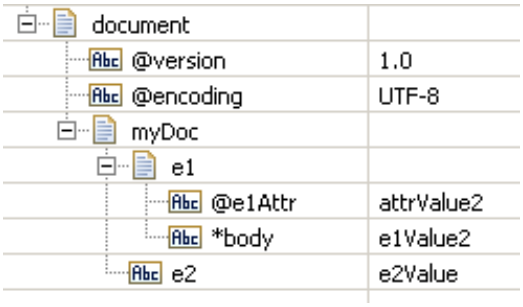
たとえば、pub.xml:xmlNodeToDocument サービスに対する入力として、次の XML ノードを指定するとします。

```
<?xml version="1.0" encoding="UTF-8"?><myDoc><e1
e1Attr="attrValue1">e1Value1</e1><e2>e2Value</e2><e1
e1Attr="attrValue2">e1Value2</e1></myDoc>
```

watt.server.xml.xmlNodeToDocument.keepDuplicates を「`true`」に設定した場合、pub.xml:xmlNodeToDocument サービスは下図のドキュメントを生成します。



watt.server.xml.xmlNodeToDocument.keepDuplicates を「false」に設定した場合、pub.xml:xmlNodeToDocument サービスは下図のドキュメントを生成します。



生成されたドキュメントではソース XML 内の最後の <e1> エレメントだけが保持されていることに注意してください。

watt.server.xml.xmlNodeToDocument.makeArrayforWS
anyType エレメントに含まれる重複エレメントを Integration Server でデコードする方法を指定します。Integration Server でタイプ anyType のエレメントに含まれる重複エレメントの配列を作成する場合は、watt.server.xml.xmlNodeToDocument.makeArrayforWS を「true」に設定します。重複エレメントを Integration Server でタイプ anyType として定義されたエレメント内で別々に繰り返されたエレメントのままにする場合は、このパラメータを「false」に設定します。「false」に設定した場合、Integration Server はタイプ anyType として定義されたエレメント内で複数回出現するエレメントの配列を作成しません。デフォルトは「false」です。

watt.server.xmlCoder.getUndefinedDataTypeClassName
サポートされていないデータタイプのクラス名が次に含まれるかどうかを決定します。

- com.wm.util.coder.XMLCoder.encode、pub.document:documentToXMLValues サービス、廃止されたサービス pub.record:XMLValuesToRecord により作成された XML ドキュメント
- pub.document:XMLValuesToDocument サービス、廃止されたサービス pub.record:XMLValuesToRecord サービスによって返された IData オブジェクト、および com.wm.util.coder.XMLCoder.decode によって返された値オブジェクト

「true」に設定すると、サポートされていないデータタイプのクラス名が、XML ドキュメント、または上記のメソッドやサービスにより作成された IData オブジェクトに含まれます。「false」に設定すると、サポートされていないデータタイプのクラス名は省略されます。デフォルトは「false」です。

watt.serverlog.

watt.serverlog.includeThreadID
サーバログに書き込まれるログメッセージにアクションを処理するスレッドのスレッド ID が含まれるかどうかを指定します。メッセージ ID の後、メッセージの前に、スレッド ID が入り、tid が接頭語として付加されます。例えば、「[ISS.0014.0002I] (tid=69) 初期化が 46 秒で完了しました。」となります。

「true」に設定すると、サーバログメッセージにスレッド ID が含まれます。「false」に設定すると、サーバログメッセージからスレッド ID が省略されます。デフォルト値は true です。

watt.ssh.

watt.ssh.jsch.ciphers
Integration Server が SFTP サーバとの通信でサポートする暗号のリストを指定します。これは、SFTP クライアントとして動作する Integration Server が SFTP サーバへの接続を確立するときに SFTP サーバに提示する暗号です。

デフォルトの値は aes256-ctr,aes192-ctr,arcfour,arcfour128,arcfour256 です。

他の暗号を含めるには、カンマ区切りリストに暗号を追加します。暗号を削除するには、リストから暗号を削除します。暗号に脆弱性が認識された場合、その暗号の削除が必要なことがあります。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.ssh.jsch.kex
SFTP サーバエイリアスの作成または編集時に [**優先キー交換アルゴリズム**] リストに表示されるキー交換アルゴリズムのカンマ区切りリストを指定します。脆弱と見なされていないキー交換アルゴリズムのみ指定します。脆弱と見なされているキー交換アルゴリズムは指定しないでください。[**優先キー交換アルゴリズム**] リストには、設定パラメータ値でアルゴリズムが指定されているのと同じ順序でキー交換アルゴリズムが表示されます。デフォルトは “” です。つまり、Integration Server は [**優先キー交換アルゴリズム**] リストにすべてのキー交換アルゴリズムを表示します。

> **重要:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.ssh.jsch.logging
JSch ログを有効にします。このプロパティを「`true`」に設定した場合、JSch からのログがすべて Integration Server コンソールにプリントされます。デフォルトは「`false`」です。

watt.ssh.jsch.mac_c2s
クライアントからサーバへの転送に対するメッセージ認証コード (MAC) アルゴリズムの順序を指定します。デフォルト値は、SFTP クライアントとして動作する Integration Server がサポートする MAC アルゴリズムのデフォルト順序です。クライアントからサーバへの転送に対する MAC アルゴリズムの使用を無効にするには、watt.ssh.jsch.mac_c2s 値から削除します。

> **注:**
> このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.ssh.jsch.mac_s2c
サーバからクライアントへの転送に対するメッセージ認証コード (MAC) アルゴリズムの順序を指定します。デフォルト値は、SFTP クライアントとして機能する Integration Server がサポートする MAC アルゴリズムのデフォルト順序です。サーバからクライアントへの転送に対する MAC アルゴリズムの使用を無効にするには、watt.ssh.jsch.mac_s2c 値から削除します。

> **注:**

このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

注:

- `watt.ssh.jsch.logging` を除くすべての `watt.ssh.jsch.*` パラメータは廃止されました。優先キー交換アルゴリズム、Ciphers、および MAC アルゴリズムはユーザインタフェースで設定されるため、廃止されたパラメータは使用しないでください。
- Integration Server は、SFTP クライアントの両バージョンのログを有効にするために、`watt.ssh.jsch.logging` サーバ設定プロパティを使用します。

watt.ssl.

watt.ssl.accelerator.provider
Solaris 10 OS マシン上の T1/T2 プロセッサに付属の SSL アクセラレータの使用を有効にします。このパラメータで指定できる値は「SunPKCS11-Solaris」だけです。このアクセラレータを使用するには、JVM 1.5 で Integration Server を実行し、[**セキュリティ**] > [キーストア] > [キーストアエイリアスの作成] ページの [HSM ベースのキーストア] フィールドを「true」に設定する必要があります。

このパラメータを指定しなかった場合、SSL ポートでは nCipher アクセラレータだけを使用できます。このアクセラレータを使用するには、[**セキュリティ**] > [キーストア] > [キーストアエイリアスの作成] ページの [HSM ベースのキーストア] フィールドを「true」に設定する必要があります。

watt.ssl.entrust.toolkit.ssl.fragmentblockcipher
ブロック暗号を使用している場合に、Entrust ライブラリで SSL レコードをフラグメント化するかどうかを指定します。

Integration Server に含まれている Entrust ライブラリは、US-CERT Vulnerability Note VU#864643 (http://www.kb.cert.org/vuls/id/864643) で示されている SSL 脆弱性に対処します。AES などのブロック暗号を使用する場合は、Entrust ライブラリによって、SSL レコードがフラグメント化される方法が変更されます。ブロック暗号を使用すると、SSL レコードは、Entrust ライブラリによって、1 バイトのレコードおよび残りのバイトで構成されるレコードの 2 つのレコードに分割されます。この変更によって、悪用の効果が失われます。

相互運用性を確保するために、SSL レコードのフラグメント化を無効にする必要がある場合は、watt.ssl.entrust.toolkit.ssl.fragmentblockcipher パラメータを「false」に設定すると、フラグメント化機能を無効にできます。このプロパティのデフォルト値は「true」です。

watt.ssl.iaik.debug
SSL クライアントと SSL サーバの間の SSL ハンドシェイク通信メッセージを Integration Server がサーバコンソールに記録するかどうかを示します。「`true`」に設定すると、Integration Server は SSL ハンドシェイク通信メッセージをサーバコンソールに記録します。デフォルトは「`false`」です。

重要:
このパラメータの設定を変更した場合、変更を適用するには、Integration Server を再起動する必要があります。

watt.tx.

watt.tx.defaultTTLMins
送信する保証付きデリバートランザクションに対してデフォルトの TTL (time-to-live : 廃棄までの時間) の値を指定します。送信トランザクションを開始するサービスが TTL 値を指定しない場合に、サーバが送信トランザクションをジョブストア内で管理する時間を分で指定します。デフォルトは `30` です。

watt.tx.disabled
送信トランザクションに対して保証付きデリバーの使用を無効にするかどうかを指定します。デフォルトで、サーバは送信トランザクションに対して保証付きデリバーの使用を許可します。デフォルトは「`false`」です。

watt.tx.heuristicFailRetry
ヒューリスティックな障害発生後に Integration Server が再起動したときに、ペンディングになっているジョブストア内の保証付きデリバートランザクションに対して Integration Server でサービスを再実行するかどうかを指定します。トランザクション状態がペンディングの場合は、Integration Server で障害が発生する前にサービスの実行が開始されています。

watt.tx.heuristicFailRetry 設定を「`true`」にすると、Integration Server はトランザクション状態を「ペンディング (Pending)」から「新規 (New)」にリセットします。また、Integration Server はサービスを再試行します。設定が「`true`」である場合は、Integration Server がサービスを実行する前にトランザクションの期限が切れたときにのみ、サービスの実行要求が失敗します。

設定を「`false`」にすると、Integration Server はトランザクション状態を「ペンディング (Pending)」からヒューリスティックな障害を示す「失敗 (Fail)」にリセットします。Integration Server はサービスを再試行しません。設定が「`false`」である場合は、ヒューリスティックな障害が発生したとき、またはトランザクションの期限が切れたときにサービスの実行要求が失敗します。

デフォルトは「`true`」です。

watt.tx.jobThreads
送信する保証付きデリバージョブストアのペンディング要求を処理するためのスレッドプール内で使用できるクライアントスレッド数を指定します。デフォルトは 5 です。

watt.tx.retryBackoffTime
サービス要求が失敗した後に Job Manager がサービスを実行する要求を Integration Server に再送信するまで待機する秒数を指定します。デフォルトは `60` です。

watt.tx.sweepTime
送信保証付きデリバートランザクションのジョブストアをスイープする間隔を秒数で指定します。サーバは、送信が必要なトランザクションを識別するためにジョブストアをスイープします。デフォルトは `60` です。

watt.tx.vm.id
Integration Server の ID を指定します。同じホストマシン上で複数の Integration Server が実行されていて、すべてのサーバで保証付きデリバーが使用されている場合に、このパラメータを使用します。各 Integration Server に一意の ID を指定することによって、重複する保証付きデリバートランザクション ID が作成されるのを防止できます。値は 0〜2147483647 の整数にする必要があります。デフォルトは `0` です。

watt.um

watt.um.clientLog.level
umClient.log と呼ばれる Universal Messaging クライアントログファイルに書き込まれる情報を指定します。各レベルはそのレベル以上のログエントリを出力します。有効な値は trace、debug、info、warn、error、fatal、および off です。デフォルトは「error」です。

watt.um.clientLog.level パラメータは、Integration Server Administrator の **[ログ] > [ログ設定] > [UM クライアントロガーの詳細の表示] > [UM クライアントロガーの詳細の編集]** ページの [**ログレベル**] パラメータに指定された値を表示します。

> **重要:**
> このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.um.clientLog.size
Universal Messaging クライアントログファイルの最大サイズ (MB)。このサイズに達すると、Integration Server は umClient.log.(*number*) というバックアップにファイルをロールオーバし、新しいファイルを作成します。デフォルトは 10 です。

watt.um.clientLog.size パラメータは、Integration Server Administrator の **[ログ] > [ログ設定] > [UM クライアントロガーの詳細の表示] > [UM クライアントロガーの詳細の編集]** ページの [**ログサイズ**] パラメータに指定された値を表示します。

> **重要:**
> このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.um.clientLog.fileDepth
Universal Messaging クライアントログファイルのログローリングを使用してディスクに保持するバックアップログファイルの数です (watt.um.clientLog.size プロパティを参照)。この数に達すると、最も古いファイルが削除されます。デフォルトは 2 です。

`watt.um.clientLog.fileDepth` パラメータは、Integration Server Administrator の **[ログ] > [ログ設定] > [UM クライアントロガーの詳細の表示] > [UM クライアントロガーの詳細の編集]** ページの [**ログの深さ**] パラメータに指定された値を表示します。

> **重要:**
> このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.um.clientTraceLog.folderLogSize
イベントライフサイクルログ処理のすべてのログファイルの最大サイズ (MB 単位) を指定します。つまり、 *Integration Server_directory* instances/*instanceName*/logs/umEventLifecycle にある、アーカイブを含めたすべてのログファイルの最大サイズです。デフォルトは 1024 です。

`watt.um.clientTraceLog.folderLogSize` の値は、各メッセージストアのトレースログ、および各メッセージストアに対して保持する多数のアーカイブログファイルを格納するのに十分な大きさに設定してください。

> **重要:**
> このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

`watt.um.clientTraceLog.folderLogSize` パラメータの値は、Integration Server Administrator の [**メッセージング**] > [ログ設定] > [UM クライアントロガーの詳細の表示] ページの [**トレー**

スフォルダログのサイズ] フィールドの値に関連付けられています。[**トレースフォルダログのサイズ**] フィールドを変更した場合、Integration Server を再起動する必要はありません。

watt.um.clientTraceLog.level
イベントライフサイクルログ処理で使用するトレースロガーのログレベルを指定します。トレースロガーは `watt.um.clientTraceLog.stores` に指定されたメッセージストアの詳細なログメッセージを書き込みます。以下のいずれかに設定します。

- `Off`: イベントライフサイクルログ処理はこの Integration Server に対して無効になります。これがデフォルト値です。
- `Info`: イベントライフサイクルログ処理は有効になります。トレースロガーは、`watt.um.clientTraceLog.stores` に指定されたキューまたはチャネルの情報レベルのメッセージを書き込みます。
- `Trace`: イベントライフサイクルログ処理は有効になります。トレースロガーは、`watt.um.clientTraceLog.stores` に指定されたキューまたはチャネルのトレースレベルのメッセージを書き込みます。`Trace` ログレベルは `Info` ログレベルより詳細です。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

`watt.um.clientTraceLog.level` パラメータの値は、Integration Server Administrator の [**メッセージング**] > [ログ設定] > [UM クライアントロガーの詳細の表示] ページの [**トレースストアログのレベル**] フィールドの値に関連付けられています。[**トレースストアログのレベル**] フィールドを変更した場合、Integration Server を再起動する必要はありません。

watt.um.clientTraceLog.stores
イベントライフサイクルログ処理の実行対象である、メッセージストア (具体的にはキュー名またはチャネル名) のカンマ区切りリスト。この Integration Server がメッセージを送受信するすべてのキューまたはチャネルで使用されるトレースロガーの場合、* を指定します。デフォルトは * です。

Universal Messaging のフォルダを使用するキューまたはチャネルを指定する場合、スラッシュでフォルダ名を区切ります。たとえば、パブリッシュ可能なドキュメントタイプ myFolder.mySubFolder:documentABC は、Universal Messaging チャネル名 wm/is/myFolder/mySubFolder/documentABC に相当します。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

`watt.um.clientTraceLog.stores` パラメータの値は、Integration Server Administrator の [**メッセージング**] > [ログ設定] > [UM クライアントロガーの詳細の表示] ページの [**トレースストア**] フィールドの値に関連付けられています。[**トレースストア**] フィールドを変更した場合、Integration Server を再起動する必要はありません。

watt.um.clientTraceLog.storeLogSize
イベントライフサイクルログ処理に対する個別のメッセージストアのトレースログファイルの最大サイズ (MB 単位) を指定します。つまり、 *Integration Server_directory* /instances/*instanceName*/logs/umEventLifecycle にある個別のログファイルの最大サイズです。デフォルトは 10 MB です。イベントライフサイクルログがこのサイズに達すると、UM クライアントはログのアーカイブ (.zip) を作成し、ストアの新しいライフサイクルログを開始します。

watt.um.clientTraceLog.folderLogSize の値がすべてのメッセージストアのトレースログを格納するのに十分な大きさでない場合、UM クライアントはメッセージストアのライフサイクルログのアーカイブファイルを保持しません。その代わり、UM クライアントは各ストアのトレースストアログファイルを単に循環します。

重要:
このパラメータの変更を適用するには、Integration Server を再起動する必要があります。

watt.um.clientTraceLog.folderLogSize パラメータの値は、Integration Server Administrator の [**メッセージング**] > [ログ設定] > [UM クライアントロガーの詳細の表示] ページの [**トレースストアログのサイズ**] フィールドの値に関連付けられています。[**トレースストアログのサイズ**] フィールドを変更した場合、Integration Server を再起動する必要はありません。

watt.wm.tnextdc.

watt.wm.tnextdc.configVersion
これは内部パラメータです。変更しないでください。

watt.wmcloud.

watt.wmcloud.listeners.monitoringInterval
有効な各 webMethods Cloud アカウントがオンプレミス Integration Server 上にアクティブリスナーを持っていることを確認するために、Integration Server がオンプレミスリスナーを監視する間隔をミリ秒で指定します。スレッド監視が実行していないリスナーを見つけた場合、スレッド監視はそのリスナーを開始しようとします。watt.wmcloud.listeners.monitoringInterval サーバ設定パラメータは、スレッド監視の実行間隔を決定します。このパラメータのデフォルト値は 300000 ミリ秒 (5 分) です。

watt.wmcloud.listeners.maxIdleTime
webMethods Cloud アカウントのオンプレミスリスナーが、再作成されずにアイドル状態でいることができる時間の長さをミリ秒単位で指定します。このパラメータのデフォルト値は 1800000 ミリ秒 (30 分) です。

C 環境変数

■ Integration Server と Microservices Runtime に定義された環境変数 1284

Integration Server と Microservices Runtime に定義された環境変数

Integration Server と Microservices Runtime には環境変数 (ENV 変数) が定義されています。これらの環境変数は以下の状況で使用できます。

- Integration Server または Microservices Runtime が Docker コンテナで実行される間にタスクの実行をサポートする。環境変数は、`docker run` コマンド内でイメージ名の直前に含まれています。
- コマンドラインから Integration Server または Microservices Runtime を起動する場合。
- Cloud Container への展開を含む、ACDL ベースの展開時の置換変数の値として。

  環境変数の値を置換値として使用するには、置換値を `${ENV_VARIABLE}` と指定します。ここで *ENV_VARIABLE* は定義された環境変数の名前です。存在しない環境変数を指定した場合、`${ENV_VARIABLE}` は保持され、設定で使用されます。

注:
特に指定のない限り、Microservices Runtime に使用する環境変数は、Microservices ライセンスを備えている Integration Server にも使用できます。

Integration Server または Microservices Runtime でサポートされている環境変数は、以下の表のとおりです。

環境変数	説明
`EXTERNALIZE_PACKAGES`	「true」に設定すると、Docker コンテナ内で実行している Integration Server に、起動時に以下のいずれかのディレクトリにあるパッケージをロードするよう指示します。 ■ *<HOST_DIR>/<SERVICE_NAME>*/packages ■ *<HOST_DIR>/<INSTANCE_NAME>*/packages *HOST_DIR*、*SERVICE_NAME*、および *INSTANCE_NAME* は、同じ名前の ENV 変数の値のセットです。 `SERVICE_NAME` を指定する場合、Integration Server は *<HOST_DIR>/<SERVICE_NAME>*/packages を検索します。`SERVICE_NAME` を指定しない場合のみ、Integration Server は *<HOST_DIR>/<INSTANCE_NAME>*/packages を検索します。 `EXTERNALIZE_PACKAGES` のデフォルト値は「false」です。

環境変数	説明
	パッケージディレクトリのコンテンツは、フォルダ名がパッケージ名である必要があります。*packageName* フォルダのコンテンツは、Integration Server パッケージの構造と一致する必要があります。パッケージディレクトリにあるパッケージは *.zip または *.7z. などのアーカイブファイルにできません。Integration Server パッケージの構造の詳細については、700 ページの「サーバによるパッケージ情報の保存」 を参照してください。
HOST_DIR	ファイルの書き込み先のホストマシンにある、マウントされたディレクトリへのパス。HOST_DIR パラメータと値を渡すか、またはそれを docker run コマンドで ENV 変数として設定すると、Integration Server のログと設定アーティファクトが外部化されます。
JAVA_MAX_MEM	最大ヒープサイズ。デフォルト値は 1024 MB です。*Integration Server_directory* /bin/setenv.bat|sh ファイルで Microservices Runtime に設定されている値を、ファイル自体の編集なしに上書きするには、JAVA_MAX_MEM 環境変数を使用します。 **注:** JAVA_MAX_MEM 環境変数は Microservices Runtime 専用です。
JAVA_MIN_MEM	最小ヒープサイズ。デフォルト値は 256 MB です。*Integration Server_directory* /bin/setenv.bat|sh ファイルで Microservices Runtime に設定されている値を、ファイル自体の編集なしに上書きするには、JAVA_MIN_MEM 環境変数を使用します。 **注:** JAVA_MIN_MEM 環境変数は Microservices Runtime 専用です。
PERSIST_LOGS	「true」に設定すると、Docker コンテナ内で実行されている Integration Server はログファイルを *HOST_DIR*/*SERVICE_NAME*/logs にパーシストします。SERVICE_NAME が指定されていない場合、*SERVICE_NAME* は *INSTANCE_NAME* に置き換えられます。Integration Server は *Integration Server_directory* /instances/<*instanceName*>/logs および profiles/IS_<*instanceName*>/logs ディレクトリにあるログを外部化します。「false」に設定すると、Docker コンテナ内で実行されている Integration

環境変数	説明
	Server はログファイルを外部化しません。デフォルトは「true」です。
PERSIST_CONFIGS	「true」に設定すると、Docker コンテナ内で実行されている Integration Server は設定ファイルを *HOST_DIR*/*SERVICE_NAME*/config ディレクトリにパーシストします。SERVICE_NAME が指定されていない場合、*SERVICE_NAME* は *INSTANCE_NAME* に置き換えられます。「false」に設定すると、Docker コンテナ内で実行されている Integration Server は設定ファイルを外部化しません。デフォルトは「true」です。
SAG_IS_AUDIT_STDOUT_LOGGERS	コンソール (STDOUT) に書き込む監査ロガーを指定します。監査ロガーが STDOUT に書き込む場合、補助出力と見なされ、ロガーは指定されたファイルまたはデータベースのログ記録先に引き続き書き込みます。以下のいずれかに設定します。 ■ STDOUT に書き込む監査ロガーのカンマ区切りリストたとえば、「WMSESSION,WMERROR」と入力します。 監査ロガー名は監査ログファイル名の最初の部分です。監査ログファイルの一覧については、『*webMethods Audit Logging Guide*』を参照してください。 ■ ロガー名のカンマ区切りリストは、Integration Server Administrator の [**ログ**] > [ログ設定] ページに表示されます。例:セッションロガー、エラーロガー ■ ALL は、すべてのファイルベースの監査ロガーを STDOUT に書き込みます。 ■ NONE は、どのファイルベースの監査ロガーも STDOUT に書き込まないことを示します。これがデフォルト値です。 変数コンテンツの大文字小文字は区別されません。 STDOUT に監査ログエントリを書き込むと、Integration Server はロガー名をエントリに含めることでエントリのソースを特定します。 Docker コンテナ内で実行している Integration Server と SAG_IS_AUDIT_STDOUT_LOGGERS 環境変数の使用の詳

環境変数	説明
	細については、1022ページの「コンソールへのサーバログの書き込み」を参照してください。 STDOUT に書き込まれる監査ログエントリのフィールドおよびレコードに使用される区切り文字は、それぞれ watt.server.audit.stdout.fieldDelimiter および watt.server.audit.stdout.recordDelimiter サーバ設定パラメータを使用して設定できます。
SAG_IS_CONFIG_PROPERTIES	Microservices Runtime またはライセンスされた Microservices Runtime 機能を持つ Integration Server で使用する設定変数テンプレートの場所と名前を指定します。
SAG_IS_CONFIG_VARIABLES_DEBUG	「true」に設定すると、Microservices Runtime が設定変数の初期化時に生成されたログに、[Debug] レベルメッセージを含めるよう指示します。「false」に設定すると、重大度が「Information」以上の設定ログメッセージが設定変数ログに含まれます。デフォルトは「false」です。 設定変数ログの詳細については、『*Developing Microservices with webMethods Microservices Runtime*』を参照してください。
SAG_IS_LICENSE_FILE	*Integration Server_directory* /instances/*instanceName*/config にある licenseKey.xml の代わりに Integration Server で使用するライセンスファイルを指定します。指定されたライセンスキーは有効なライセンスファイルであり、Integration Server によってアクセス可能な場所にある必要があります。Docker コンテナで実行している Integration Server に指定されたライセンスキーは、ホストファイルシステムのマウントされたディレクトリで使用可能である必要があります。起動時に、SAG_IS_LICENSE_FILE によって指定された場所にファイルが存在する場合、Integration Server はそのファイルを指定された場所から *Integration Server_directory* /instances/*instanceName*/config にコピーし、ファイルを licenseKey.xml として保存します。 起動時にライセンスファイルを指定すると、Integration Server の Docker イメージに期限切れのライセンスファイルが含まれている場合や、既存の Docker イメージで代替のライセンスファイルを指定したい場合に役に立ちます。SAG_IS_LICENSE_FILE を使用してライセンスファイルを指定することにより、

環境変数	説明
	Docker イメージを再作成することなくライセンスファイルを変更できます。
SAG_IS_HEALTH_ENDPOINT_ACL	メンバーがヘルスゲージエンドポイント /health へのアクセスおよびそのエンドポイントの呼び出しをできるアクセスコントロールリスト (ACL) を指定します。 is_container.bat/sh スクリプトを使用して作成された Docker コンテナで実行される Microservices Runtime または Integration Server の場合、ヘルスゲージエンドポイントにアクセスするためのデフォルトの ACL は「Anonymous」です。これは、エンドポイントが呼び出される場合に認証は必要ないことを意味します。他のすべての Microservices Runtime または Integration Server の場合、デフォルト ACL は「Administrator」です。
SAG_IS_METRICS_ENDPOINT_ACL	メンバーがメトリクスエンドポイント /metrics へのアクセスおよびそのエンドポイントの呼び出しをできるアクセスコントロールリスト (ACL) を指定します。 is_container.bat/sh スクリプトを使用して作成された Docker コンテナで実行される Microservices Runtime または Integration Server の場合、メトリクスエンドポイントにアクセスするためのデフォルトの ACL は「Anonymous」です。これは、エンドポイントが呼び出される場合に認証は必要ないことを意味します。他のすべての Microservices Runtime または Integration Server の場合、デフォルト ACL は「Administrator」です。
SECRET_PATH	コンテナ内の Kubernetes Secret へのパスを指定します。docker run コマンドで SECRET_PATH ENV 変数を渡すと、Kubernetes Secret に保存された機密データを設定変数テンプレートから参照できます。
SERVICE_NAME	Integration Server アーティファクトをパーシストする HOST_DIR の下の一意のディレクトリ名。指定すると、logs および config ディレクトリが *HOST_DIR*/*SERVICE_NAME* ディレクトリの下に作成され、logs および config ディレクトリのすべてのアーティファクトはそれぞれのディレクトリでパーシストされます。*SERVICE_NAME* を指定しない場合、*HOST_DIR* ディレクトリの下に作成される一意のディレクトリ名として Integration Server インスタンス名

環境変数	説明
	が使用されます。各 Docker コンテナに一意のディレクトリを指定することをお勧めします。

D FIPS 140-2 準拠

■ FIPS 140-2 準拠 1292

FIPS 140-2 準拠

webMethods Integration Server バージョン 9.0 以降には、Entrust Authority Security Toolkit for Java 8 および BouncyCastle ライブラリが組み込まれています。このツールキットは、FIPS 140-2 認定を取得しています。FIPS (米連邦情報処理規格) は、米連邦政府の内部で使用する情報処理規格です。バージョン 8 用のポリシーは次のページから入手できます。

http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2012.htm#1839

多くの政府機関や金融機関では、ソフトウェアが FIPS 140-2 準拠であり、現時点での暗号化情報処理の基準およびガイドラインに従っていることを要求しています。

注:
Integration Server 自体は、FIPS 140 認定であると見なされていません。

Integration Server を FIPS 140-2 準拠モードで実行すると、必ず FIPS 準拠モードの FIPS 準拠アルゴリズムのみが使用されます。FIPS モードを有効にするには、Integration Server で以下の拡張設定を行います。

watt.security.fips.mode=true

このサーバ設定パラメータの詳細については、1095 ページの「サーバ設定パラメータ」を参照してください。また、Integration Server の拡張設定の表示および更新方法については、146 ページの「拡張設定の使い方」を参照してください。

FIPS 準拠モードでのサーバ実行のほか、以下の Entrust Cryptographic Module セキュリティポリシーおよび BouncyCastle FIPS 準拠セキュリティポリシーの手順にも従う必要があります。この手順は、コンピュータへの複数ユーザのアクセス禁止、コンピュータが物理的に保護されていることの確認などの保護対策を実施する手順です。この情報はドキュメントの「*Operational Environment*」セクションで入手できます。組織のポリシーに応じて、Entrust および BouncyCastle 認可で使用されたものと同じハードウェア、オペレーティングシステムおよび JDK の使用が必要になる場合もあります。

FIPS モード暗号化は、HTTPS または FTPS 通信と S/MIME 暗号化/署名にのみ適用できます。FIPS Entrust 暗号化/署名は HTTPS または FTPS に使用され、FIPS BouncyCastle 暗号化/署名は S/MIME 通信に使用されます。

E Integration Server がクライアントとして機能するときの NTLM 認証の使用

- 概要 1294
- Java ベースの NTLM サポートの使用 1294
- 統合 Windows 認証によるネイティブ NTLM サポートの使用 1295

概要

Integration Server で Web ページにアクセスするサービスを実行すると、Integration Server は Web サーバに要求を発行する Web クライアントのように動作します。Integration Server では、Integration Server から NTLM (Windows NT LAN Manager) 認証をサポートする Web サーバへの接続で、NTLM 認証をサポートしています。NTLM クライアントサポート用に正しく設定された場合、Integration Server は適切な認証クレデンシャルを持つ Web サーバからの NTLM チャレンジに応答します。

Integration Server を使用して、Integration Server がクライアントとして機能するサービス要求に対する NTLM 認証のサポートを有効にできます。Integration Server は NTLM クライアントサポートの 2 つのバリアントを備えています。

- UNIX プラットフォームおよび Windows システム用の Java ベースの NTLM
- Windows システム用のネイティブ NTLM

  ネイティブ NTLM の場合、Integration Server は Integration Server とイントラネット上の Web サーバ間の接続を確立する間にその ID を認証する方法として統合 Windows 認証を使用します。

この付録では、Integration Server がクライアントとしてのみ機能するときに Integration Server で NTLM 認証を使用する方法について説明します。

Java ベースの NTLM サポートの使用

デフォルトで、Integration Server は Java ベースの NTLM サポートを使用します。Java ベースの NTLM サポートは UNIX および Windows プラットフォームで使用できます。

Integration Server が Windows、UNIX または別のサポートされるプラットフォームのいずれで実行されているのかに関係なく、Integration Server は、適切な認証クレデンシャルを持つ Web サーバからの NTLM チャレンジに応答します。

Integration Server における Java ベースの NTLM 認証には次の制限があります。

- Java ベースの NTLM 認証は NTLMv2 のみサポートします。
- Java ベースの NTLM 認証は NTLMv1 をサポートしません。
- Java ベースの NTLM 認証は HTTP および HTTPS で使用できます。
- Integration Server における Java ベースの NTLM 認証は NTLM プロキシサーバでは動作しません。

Java ベースの NTLM クライアント認証を使用する場合、以下の情報に注意してください。

- 認証クレデンシャルは明示的に指定する必要があります。認証情報を指定するときに、ドメイン名とそれに続くバックスラッシュ (\) をユーザ名の前に付ける必要があります。たとえば、pub.client:http サービスの呼び出しの認証タイプとして NTLM を使用するときは、*domain_name**user_name* の形式を使用してユーザ入力パラメータの値を指定する必要があります。

pub.client:http サービスにおける *auth* タイプの設定の詳細については、『*webMethods Integration Server Built-In Services Reference*』を参照してください。

- NTLM サーバは NTLM ヘッダーおよび Negotiate ヘッダーを送信するよう設定する必要があります。Integration Server が Negotiate ヘッダーのみ受信する場合、NTLM ハンドシェイクは実行されません。

注:
Java ベースの NTLM 認証を使用する場合、ネイティブ NTLM が無効なことを確認してください。デフォルトで、Integration Server は Java ベースの NTLM 認証を使用し、ネイティブ NTLM は無効になります。ネイティブ NTLM サポートを無効にする方法の詳細については、1296 ページの「統合 Windows 認証の非アクティブ化」を参照してください。

統合 Windows 認証によるネイティブ NTLM サポートの使用

Integration Server が Windows で実行される場合、Integration Server は Integration Server の ID の認証方法として「統合 Windows 認証」を使用する「ネイティブ」NTLM サポートを使用できます。統合 Windows 認証は、ネットワーク上で実際のパスワードまたはアカウントについての機密情報の伝送を要求することなく、ユーザを認証します。

統合 Windows 認証要求に応答する場合に Integration Server が使用するクレデンシャルは、Integration Server がスタンドアロンアプリケーションとして実行されているか NT サービスとして実行されているかによって異なります。

- Integration Server がスタンドアロンアプリケーションとして実行されている場合、ログインした Windows ユーザのクレデンシャルを使用します。
- Integration Server が NT サービスとして実行されている場合、認証にローカルシステム権限を使用します。ユーザとしてログインすると、Integration Server は、統合 Windows 認証に応答する際に、そのセッションに関連するクレデンシャルを使用します。

Integration Server におけるネイティブ NTLM 認証には次の制限があります。

- ネイティブ NTLM 認証は Windows システム専用です。
- ネイティブ NTLM 認証は HTTP のみサポートします。
- ネイティブ NTLM 認証は NTLM プロキシサーバにはサポートされていません。
- Integration Server がネイティブ NTLM サポートを使用するには、Web サーバが統合 Windows 認証をサポートする必要があります。Microsoft Internet Information Server (IIS) は統合 Windows 認証をサポートする Web サーバの一例です。

Integration Server でネイティブ NTML 認証を使用するには、最初に Integration Server で統合 Windows 認証をアクティブにする必要があります。統合 Windows 認証をアクティブにする方法の詳細については、1295 ページの「統合 Windows 認証のアクティブ化」を参照してください。

統合 Windows 認証のアクティブ化

統合 Windows 認証をアクティブにすると、Integration Server は自動的にその要求に応答するようになります。

統合 Windows 認証をアクティブにするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. WmWin32 パッケージが有効になっていない場合は、このパッケージの [**有効**] 列の [**いいえ**] をクリックして有効にします。
4. パッケージのリストで [**WmWin32**] をクリックします。

   注:
   WmWin32 パッケージは、Integration Server 7.1 の時点で廃止されています。
5. [**WmWin32 内のサービスを表示**] をクリックします。
6. サービスのリストで、[**wm.ntlm:reg**] をクリックします。
7. [**reg のテスト**] をクリックします。サーバに win32.ntlm.reg サービスのテストページが表示されます。
8. [**テスト (入力値なし)**] をクリックします。統合 Windows 認証がアクティブになります。

   注:
   Integration Server が稼動している場合に常に統合 Windows 認証を使用可能にするには、win32.ntlm:reg service を Win32 パッケージの開始サービスとして使用します。開始サービスの割り当ての詳細については、『*webMethods Service Development Help*』を参照してください。

統合 Windows 認証の非アクティブ化

統合 Windows 認証を非アクティブにするには

1. Integration Server Administrator を開いていない場合は、それを開きます。
2. [**パッケージ**] > [管理] に移動します。
3. パッケージのリストで [**WmWin32**] をクリックします。
4. [**WmWin32 内のサービスを表示**] をクリックします。
5. サービスのリストで、[**wm.ntlm:unreg**] をクリックします。
6. [**unregのテスト**] をクリックします。サーバに win32.ntlm.unreg サービスのテストページが表示されます。

7. [**テスト (入力値なし)**] をクリックします。統合 Windows 認証が非アクティブになります。

F Integration Server からのユーザデータの削除

- ユーザデータの削除 1300
- ユーザアカウントへの参照の削除 1300
- 電子メールアドレスへの参照の削除 1303
- ログファイルからの個人データの削除 1304

ユーザデータの削除

GDPR (General Data Protection Regulation: EU一般データ保護規則) などのデータ保護の法律や規則では、ユーザプロファイルを削除した後でも、特定のユーザデータの取り扱いが必要となる場合があります。また、Integration Server にユーザアカウントを持つ従業員または他のクライアントが、ユーザ名、電子メールアドレス、クライアント IP アドレスなどのユーザ識別情報を Integration Server から削除するよう要求することがあります。データ保護要求およびユーザの要求に従うために、ユーザアカウントを削除すると共に、認証マッピングの変更、サーバ設定パラメータ値の修正、サービスへの新しい実行ユーザの割り当て、ログファイルの編集などのようなアクティビティを完了する必要があることがあります。

ユーザアカウントへの参照の削除

ユーザアカウントを削除する前に、実行ユーザ、認証マッピング、送信接続設定など、ユーザアカウントを参照する Integration Server のすべての場所を更新する必要があります。ユーザアカウントに依存する機能を更新する前にそのユーザアカウントを削除した場合、Integration Server に障害が発生することがあります。

Software AG では、ユーザ関連の更新を次の順序で完了することをお勧めします。

1. クライアント認証マッピングを変更します。インポートされたクライアント認証または CA (認証局) 署名用認証は、ユーザアカウントにマップされます。クライアント認証にマップされたユーザアカウントを削除する前に、以下のいずれかを実行します。

   - 認証マッピングを変更します。詳細については、554 ページの「認証マッピングの変更」を参照してください。
   - ISInternal JDBC プールのデータベーステーブル IS_CERTIFICATE_MAP に保存された認証マッピングを削除します。ユーザは複数の認証にマップできるため、複数のマッピングの削除が必要なことがあります。次のデータベース DELETE ステートメントを使用すると、ユーザ認証マッピングを迅速に削除できます。

     ```
     DELETE FROM IS_CERTIFICATE_MAP WHRE CERT_USER = “username”
     ```

     username は削除するユーザアカウントです。

2. タスク、サービス、トリガーを実行するために割り当てられたユーザアカウントを変更します。以下の 1 つ以上の更新が必要なことがあります。

更新するアセット	具体的方法	使用ツール
スケジュール済みタスク	[**サーバ**] > [スケジューリング] > [ユーザタスク] > [タスクの更新] ページの [**実行ユーザ**] の値	Integration Server Administrator
JMS トリガー	トリガーの [実行ユーザ] プロパティ	Designer

更新するアセット	具体的方法	使用ツール
Universal Messaging からのメッセージを受信する webMethods messaging trigger	トリガーの [実行ユーザ] プロパティ	Designer
Broker からのメッセージまたはローカルにパブリッシュされたメッセージを受信する webMethods messaging trigger	[設定] > [リソース] > [ストアの設定] ページで利用できる [トリガーサービスを実行するユーザ] フィールド	Integration Server Administrator
サービスをカスタムフィルタとして使用する Enterprise Gateway ルール	[セキュリティ] > [Enterprise Gateway ルール] > [ルール] > *rulename* > [編集] ページのカスタムフィルタの [実行ユーザ] フィールド	Enterprise Gateway Server として機能する Integration Server 上の Integration Server Administrator
フローサービスを呼び出してルール違反を通知する Enterprise Gateway 警告	[セキュリティ] > [Enterprise Gateway ルール] > [デフォルト警告オプションの編集] ページのデフォルト警告プロパティの [実行ユーザ] フィールド	Enterprise Gateway Server として機能する Integration Server 上の Integration Server Administrator
電子メールポート	[**サーバ**] > [ポート] > [電子メールクライアント設定の編集] ページの [ユーザを指定してサービスを実行] フィールド	Integration Server Administrator
ファイルポーリングポート	[**サーバ**] > [ポート] > [ファイルポーリングの設定の編集] ページの [ユーザを指定してサービスを実行] フィールド	Integration Server Administrator
現在のサーバのパッケージサブスクリプション	[パッケージ] > [パブリッシュ] > [サブスクライバの編集] ページの [リモートユーザ名] フィールド	Integration Server Administrator
リモートサーバのパッケージサブスクリプション	[パッケージ] > [サブスクライブ] > [サブスクライバの編集] ページの [ローカルユーザ名] フィールド	Integration Server Administrator
WmCloud アカウント設定	[webMethods Cloud] > [アカウント] > [アカウントの編集] ページの [実行ユーザ] フィールド	Integration Server Administrator

注:
上記のリストには、アダプタに割り当てられた実行ユーザは含まれていません。

3. 以下を含む、ユーザを値として指定するサーバ設定パラメータを変更します。

   - watt.server.cache.prefetchUser
   - watt.server.event.routing.runAsUser
   - watt.server.eventHandlerUser

   Integration Server Administrator の [拡張設定] ページを使用して、サーバ設定パラメータを編集します。

   注:
   サーバ設定パラメータの上記リストは完全なものではなく、修正によってまたはアダプタなどの階層化された製品によって追加されたパラメータが含まれていないことがあります。

4. 送信接続設定を変更します。送信接続を確立するため、サービスを含め、ユーザ名が指定されている Integration Server の場所。以下のものが含まれることがあります。

   - リモートサーバエイリアス。Integration Server Administrator により編集できます。
   - Web サービスエンドポイントエイリアス。Integration Server Administrator の[設定] > [Web サービス] > *endpointAliasName* > [編集] ページから編集できます。
   - 以下を含む、メッセージング接続エイリアス。
     - JMS 接続エイリアス。Integration Server Administrator の [**メッセージング**] > [JMS 設定] > [JMS 接続エイリアス] > [JMS 接続エイリアスの編集] ページから編集できます。
     - webMethods メッセージング接続エイリアス。Integration Server Administrator の [**メッセージング**] > [webMethods の設定] > [Universal Messaging 接続エイリアス] > [Universal Messaging 接続エイリアスの編集] ページから編集できます。
   - 送信 HTTP 呼び出し (pub.client:http、pub.client:soapClient または Web サービスコネクタを使用したものなど)。

5. 設定変数テンプレートを変更します。Docker コンテナ内で実行している Microservices Runtime イメージに使用されている設定変数テンプレート、またはオンプレミスの Microservices Runtime は、キーと値のペアの 1 つにユーザ名を指定します。テキストエディタを使用して、テンプレートを編集し Microservices Runtime から削除するユーザ名を指定するプロパティキーの値を変更します。ユーザ名のプロパティキーには、通常「ユーザ」または「プリンシパル」が含まれます。

   注:
   Microservices Runtime の Docker イメージが application.properties テンプレートを含む場合、イメージから作成された各 Docker コンテナもテンプレートを含みます。Docker コンテナが破壊されると、テンプレートは削除されます。コンテナが定期的に再利用されない場合、コンテナファイルシステムに添付し、application.properties テンプレートからユーザ名をスクラブします。

   注:

設定変数テンプレートはデフォルトでは Microservices Runtime に含まれます。Microservices Runtime ライセンスを備えている Integration Server も設定変数機能を使用できます。

6. Integration Server からユーザを削除します。
7. ユーザ名が存在する各 Integration Server に対して手順 2～7 を繰り返します。たとえば、Integration Server のクラスタを使用する場合、クラスタのすべてのサーバに対して手順を繰り返す必要があります。

注:
クライアント認証マッピングは、クラスタで共有されたデータベースに保存されます。クラスタのすべての Integration Server に対して手順 1 を繰り返す必要はありません。

8. 影響を受ける Integration Server インスタンス用の Docker イメージを再作成します。

電子メールアドレスへの参照の削除

Integration Server からユーザへの参照を削除する手順の一部として、電子メールアドレスへの参照の削除が必要なこともあります。以下の内容を確認し、必要に応じて電子メールアドレスを削除するか置き換えます。

- 重要な情報、新しいパッケージリリース、ルール違反などに関する通知に対する電子メールアドレス。以下の 1 つ以上に対して電子メールアドレスの更新が必要なことがあります。

更新するアセット	具体的方法	使用ツール
電子メール通知の設定	[設定] > [リソース] > [リソースの設定の編集] ページの [内部用電子メール] フィールドおよび [サービス用電子メール] フィールド	Integration Server Administrator
パスワードの有効期限通知用の電子メールアドレス	[セキュリティ] > [ユーザ管理] > [パスワードの有効期限の設定] > [編集] ページの [有効期限を通知する電子メールアドレス] フィールド	Integration Server Administrator
電子メールを送信してルール違反を通知する Enterprise Gateway 警告	[セキュリティ] > [Enterprise Gateway ルール] > [デフォルト警告オプションの編集] の [電子メール ID] フィールド	Enterprise Gateway Server として機能する Integration Server 上の Integration Server Administrator
この Integration Server のパッケージサブスクリプションの通知先電子メールアドレス	[パッケージ] > [パブリッシュ] > [サブスクライバの編集] ページの [通知先電子メールアドレス] フィールド	Integration Server Administrator

更新するアセット	具体的方法	使用ツール
リモート Integration Server のパッケージサブスクリプション用通知電子メールアドレス	[パッケージ] > [サブスクライブ] > [サブスクライバの編集] ページの [通知先電子メールアドレス] および [自動プル用電子メールアドレス]	Integration Server Administrator

- 電子メールアドレスを値として指定するサーバ設定パラメータ。
  - watt.server.email.from
  - watt.server.errorMail
  - watt.server.serviceMail
  - watt.server.txMail

  注:
  サーバ設定パラメータの上記リストは完全なものではなく、修正によってまたはアダプタなどの階層化された製品によって追加されたパラメータが含まれていないことがあります。

  Integration Server Administrator の [**拡張設定**] ページを使用して、サーバ設定パラメータを編集します。
- pub.client:smtp service、pub.replicator.notifyPackageRelease サービス、または電子メールアドレスを入力パラメータとして使用する他のすべてのサービスを呼び出すサービス。

注:
Integration Server で使用される電子メールアドレスの削除または更新を行った後、影響を受ける Integration Server インスタンス用の Docker イメージを再作成する必要があります。

ログファイルからの個人データの削除

要求により、Integration Server ログファイルからユーザを識別する個人情報 (PII) の削除が必要なことがあります。このデータには、ユーザ名、電子メールアドレス、クライアント IP アドレスが含まれていることがあります。ここでは、各ログファイルの PII に関する情報、データの検索方法、および削除方法について説明します。

注:
Integration Server を更新してユーザアカウントと電子メールアドレスへの参照を削除し、ユーザアカウントを削除した後にのみ、ログファイルから PII データを削除します。

サーバログからの個人情報の削除

Integration Server サーバログには、Integration Server で発生する処理とエラーに関する情報が含まれます。Integration Server および他の階層化された製品でサーバログに書き込むデータの量は、使用するログレベルによって異なります。

最小限の場合、Integration Server ではサーバログを毎日循環しますが、サーバログの循環はサイズに基づいて設定可能なため、1 日で複数のログが作成されることもあります。Integration Server が server.log ファイルを循環する場合に、Integration Server は現在のログの名前をアーカイブファイル名を使用するように変更して、新しい server.log ファイルを起動します。アーカイブファイル名は server.log_*yyyyMMdd_HHmmssSSSZ* の形式を使用します。*yyyyMMdd_HHmmssSSSZ* はログファイルが作成された日時です。Integration Server では、サーバログとサーバログアーカイブファイルを同じディレクトリに保存します。

サーバログのデフォルト名と場所は *Software AG_directory* \ *Integration Server_directory* \instances*instance_name*\logs\server.log です。

次の表に、server.log に書き込まれる可能性があるユーザデータの種類、検索方法および削除方法を示します。

データ	検索方法、削除方法
ユーザ名	server.log またはアーカイブファイルでユーザ名の検索と置き換えを実行するには、テキストエディタを使用します。たとえば、server.log ファイルでユーザ名を検索し、そのユーザ名を匿名文字列または空白文字列に置き換えることができます。
クライアント IP アドレス	Integration Server のサーバログメッセージにクライアント IP アドレスが含まれることはあまりありません。このメッセージを検索するには、テキストエディタを使用し、サーバログファイルで次のメッセージ ID を持つメッセージを検索し、メッセージのクライアント IP アドレスを削除するか置き換えます。 ISS.0053.0002C ISS.0053.0012C ISS.0053.0017C ISS.0138.0505E ISC.0037.0013D ISC.0064.0014T ISC.0064.0015T ISC.0064.0020T ISC.0064.0021T ISC.0064.0029T
電子メールアドレス	Integration Server の server.log メッセージに電子メールアドレスが含まれることはあまりありません。server.log またはアーカイブファイルで電子メールアドレスの検索と置き換えを実行するには、テキストエディタを使用します。

データ	検索方法、削除方法
	Integration Server では、メッセージ ID ISC.0063.0070D のメッセージについて、管理者の電子メールアドレスをログに記録します。

注:
Docker コンテナで Integration Server の Docker イメージを実行すると、Integration Server はサーバログを server.log ファイルだけでなくコンソールにも書き込みます。Docker で使用されるログドライバによって生成されたログファイルの編集が必要なことがあります。たとえば、Docker のデフォルトログドライバである json-file ログドライバは、STDOUT に書き込まれるすべての情報を取得し、JSON ファイルに書き込みます。JSON ログファイルは循環されることがあります。Docker コンテナが破棄されると、JSON ログファイルは削除されます。その結果、JSON ログファイルから識別ユーザデータを削除する必要はなくなる場合があります。ただし、Docker ボリュームを使用して、ホストファイルシステムのマウントされたディレクトリに設定ファイルとログファイルをパーシストした場合、上記のように外部化したログファイルの消去が必要なことがあります。

セッションログからの個人情報の削除

セッションログのセッション監査レコードには、Integration Server に接続するユーザのユーザ名とクライアント IP アドレスが含まれます。そのデータの場所および削除方法は、セッションログの記録先がファイルであるかデータベースであるかによって異なります。

セッションログの記録先	ユーザデータの検索方法と削除方法
ファイル	ファイルの場所: *Software AG_directory* \ *Integration Server_directory* \instances*instance_name*\logs ファイル名: *WMSESSION_serverStartupTimestamp*.log ユーザ ID の長さは 64 文字です。ログの 528 列目から記録されます。 ユーザが接続に使用した IP アドレスは 128 文字です。ログの 593 列目から記録されます。 ユーザ ID およびクライアント IP アドレスの検索と置き換えを行うには、テキストエディタを使用します。たとえば、セッションログファイルでユーザ名を検索し、そのユーザ名を匿名文字列または空白文字列に置き換えることができます。
データベース	セッション監査ロガーが ISCoreAudit 機能エイリアスで識別される外部データベースにログを記録するよう設定されている場合、Integration Server はログメッセージを ISCoreAudit データベースの WMSESSION テーブルに書き込みます。 USERID 列にはユーザ ID が含まれます。 SESSIONNAME 列には、ユーザが接続に使用する IP アドレスが含まれます。

セッションログの記録先	ユーザデータの検索方法と削除方法

	次の SQL ステートメントを使用してデータを削除します。
	`DELETE FROM WMSESSION WHERE USERID = 'username'`
	username はログから削除するユーザアカウントです。
	`DELETE FROM WMSESSION WHERE SESSIONNAME = 'ipAddress'`
	ipAddress はログから削除するクライアント IP アドレスです。

サービスログからの個人情報の削除

サービスログのサービス監査ログレコードには、サービスを実行するユーザのユーザ名が含まれます。そのデータの場所および削除方法は、サービスログの記録先がファイルであるかデータベースであるかによって異なります。

サービスログの記録先	ユーザデータの検索方法と削除方法
ファイル	ファイルの場所: *Software AG_directory* \ *Integration Server_directory* \instances*instance_name*\logs ファイル名: *WMSERVICE_serverStartupTimestamp*.log ユーザ ID の長さは 64 文字です。ログの 1024 列目から記録されます。 ユーザ ID の検索と置き換えを行うには、テキストエディタを使用します。たとえば、サービスログファイルでユーザ名を検索し、そのユーザ名を匿名文字列または空白文字列に置き換えることができます。
データベース	サービス監査ロガーが ISCoreAudit 機能エイリアスで識別される外部データベースにログを記録するよう設定されている場合、Integration Server はログメッセージを ISCoreAudit データベースの WMSERVICE テーブルに書き込みます。 USERID 列にはユーザ ID が含まれます。 次の SQL ステートメントを使用してデータを削除します。 `DELETE FROM WMSERVICE WHERE USERID = 'username'` *username* はログから削除するユーザアカウントです。

セキュリティログからの個人情報の削除

セキュリティログのセキュリティ監査ログレコードには、ログに記録されたアクティビティを実行するユーザのユーザ ID が含まれます。そのデータの場所および削除方法は、セキュリティログの記録先がファイルであるかデータベースであるかによって異なります。

セキュリティログの記録先	ユーザデータの検索方法と削除方法
ファイル	ファイルの場所: *Software AG_directory* \ *Integration Server_directory* \instances*instance_name*\logs ファイル名: *WMSECURITY_serverStartupTimestamp*.log ユーザ ID の長さは 64 文字です。ログの 640 列目から記録されます。 ユーザ ID の検索と置き換えを行うには、テキストエディタを使用します。たとえば、セキュリティログファイルでユーザ名を検索し、そのユーザ名を匿名文字列または空白文字列に置き換えることができます。
データベース	セキュリティ監査ロガーが ISCoreAudit 機能エイリアスで識別される外部データベースにログを記録するよう設定されている場合、ログメッセージは ISCoreAudit データベースの WMSECURITY テーブルに書き込まれます。 USERID 列にはユーザ ID が含まれます。 次の SQL ステートメントを使用してデータを削除します。 `DELETE FROM WMSECURITY WHERE USERID = 'username'`

SSL セッションログからの個人情報の削除

SSL セッションログ (inboundSSLSessions.log) のセッション情報には、Integration Server に接続するユーザのユーザ名とクライアント IP アドレスが含まれます。SSL セッションログは常に特定のファイル形式になります。次の情報を使用してユーザデータを検索します。

- ファイルの場所: *Software AG_directory* \ *Integration Server_directory* \instances*instance_name*\logs
- ファイル名: *inboundSSLSessions*.log
- ユーザ ID: ユーザ ID の長さは 64 文字です。ログの 21 列目から記録されます。
- クライアント IP アドレス: ユーザが接続に使用した IP アドレスは 128 文字です。ログの 24 列目から記録されます。

ユーザデータを削除するには、テキストエディタを使用して検索を行い、ユーザ ID およびクライアント IP アドレスを置き換えます。たとえば、セッションログファイルでユーザ名を検索し、そのユーザ名を匿名文字列または空白文字列に置き換えることができます。

設定変数ログからの個人情報の削除

設定変数ログには、Microservices Runtime が起動時に設定変数テンプレートを適用する間に発生する操作、警告、およびエラーが含まれます。これにはユーザ名のプロパティキーに対して特定の値を設定するさいのメッセージが含まれます。設定変数ログはオンプレミスの Microservices Runtime の起動時に上書きされます。Docker コンテナが破棄されると、Docker コンテナで実行する Microservices Runtime の設定変数ログも破棄されます。

Microservices Runtime は、設定変数ログメッセージをコンソール (STDOUT) または以下の場所、もしくはその両方に書き込みます。 *Integration Server_directory* /instances/*instanceName*/logs/configurationvariables.logDocker コンテナ内で実行している Microservices Runtime は常に設定変数ログメッセージを両方の場所に書き込みます。

ユーザ名またはユーザ ID への参照を configurationvariables.log から削除するには、テキストエディタを使用してユーザ名またはユーザ ID でログを検索します。次に、見つかったデータを削除するか置き換えます。

ユーザ名への参照を Docker コンテナの configurationvariables.log から削除するには、コンテナファイルシステムに添付し、その後検索して、ログからユーザ名を削除するか変更します。

Microservices Runtime が設定変数ログをコンソール (STDOUT) に書き込んでいる場合、STDOUT ストリームを取得している記録先から、ユーザ名情報を削除するか置き換える必要があります。

- Docker は デフォルトでは JSON ファイル内で STDOUT を取得します。Docker で使用されるログドライバによって生成されたログファイルの編集が必要なことがあります。たとえば、Docker のデフォルトログドライバである json-file ログドライバは、STDOUT に書き込まれるすべての情報を取得し、JSON ファイルに書き込みます。JSON ログファイルは循環されることがあります。Docker コンテナが破棄されると、JSON ログファイルは削除されます。その結果、JSON ログファイルから識別ユーザデータを削除する必要はなくなる場合があります。ただし、Docker ボリュームを使用して、ホストファイルシステムのマウントされたディレクトリに設定ファイルとログファイルをパーシストした場合、上記のように外部化したログファイルの消去が必要なことがあります。
- オンプレミスの Microservices Runtime では、STDOUT ストリームは *Software AG_directory* /profiles/IS_*instanceName*/logs/wrapper.log に書き込まれます。

  logs フォルダの wrapper.log およびその他のログファイルでユーザ情報を検索するには、テキストエディタを使用します。次に、見つかったデータを削除するか置き換えます。

HTTP ログからの個人情報の削除

HTTP ログには、Integration Server によって受信された HTTP/S 要求/応答ペアごとに 1 つのエントリが含まれます。各エントリには、要求の時間、要求処理時間、応答のサイズなど、要求/応答ペアに関する情報が含まれます。各エントリに記録されるデータ量は選択されたログ形式によって異なります。

Integration Server は毎日午前零時に HTTP ログを循環します。Integration Server が http.log ファイルを循環する場合に、Integration Server は現在のログの名前をアーカイブファイル名を使用するように変更して、新しい http.log ファイルを起動します。アーカイブファイルが使用する形式は http*yymmdd_hhmm*.log です。*yymmdd_hhmm* は、Integration Server がログファイ

ルを循環した日付と時刻です。Integration Server では、HTTP ログと HTTP ログアーカイブファイルを同じディレクトリに保存します。

HTTP ログの名前と場所は *Integration Server_directory* \instances*instance_name*\logs\http.log です。

次の表に、http.log に書き込まれる可能性がある使用データの種類、検索方法および削除方法を示します。

データ	検索方法、削除方法
ユーザ名	http.log またはアーカイブファイルでユーザ名の検索と置き換えを実行するには、テキストエディタを使用します。たとえば、http.log ファイルでユーザ名を検索し、そのユーザ名を匿名文字列または空白文字列に置き換えることができます。
クライアント IP アドレス	テキストエディタを使用し、HTTP ログファイルでクライアント ID アドレスを持つエントリを検索し、エントリのクライアント IP アドレスを削除するか置き換えます。

Axis2、Kerberos、SAML およびその他のサードパーティライブラリによってログに記録される個人情報の削除

Integration Server で使用されるサードパーティライブラリは、特にデバッグレベルまたはトレースログレベルでユーザデータをログに記録できます。これらのメッセージは *Software AG_directory* /profiles/IS_*instanceName*/logs フォルダに保存されるログファイルに記録されます。このディレクトリに置かれる wrapper.log では、System.out に出力されるメッセージをすべて取得します。logs フォルダの wrapper.log およびその他のログファイルでユーザ情報 (ユーザ名、クライアント IP アドレス、電子メールアドレス) を検索するには、テキストエディタを使用します。次に、見つかったデータを削除するか置き換えます。

G Integration Server でのワイヤレス通信

- 概要 1312
- Integration Server とワイヤレスデバイス間の通信方式 1312
- Integration Server へのワイヤレス接続のための URL 使用 1313
- WML と HDML のサンプル 1316

概要

webMethods Integration Server は、インターネットを使用できるワイヤレスデバイスとの間で、要求の受信および応答の送信ができます。ワイヤレスデバイスは、URL を使用して Integration Server からの情報を要求します。サーバから送信される応答は、WML (Wireless Markup Language) コンテンツまたは HDML (Handheld Device Markup Language) コンテンツで構成されます。Integration Server が通信できるワイヤレスデバイスには、インターネットを使用できる携帯電話や PDA などがあります。

次のような場合に、ワイヤレスデバイスを使用して Integration Server と通信します。

- 会社またはサプライヤの在庫状況を確認する。
- 発注または既存の受注状況の確認を行う。
- ワイヤレスデバイスでサブミットした発注に対する注文確認を受信する。
- 株価が変動したため取引を約定するように、利用者に知らせる通知を送信または受信する。
- ワイヤレスデバイスに情報を送信するイベントハンドラを使用して、Integration Server の統計情報を収集する。
- Integration Server に保存された HDML または WML ページを要求する。

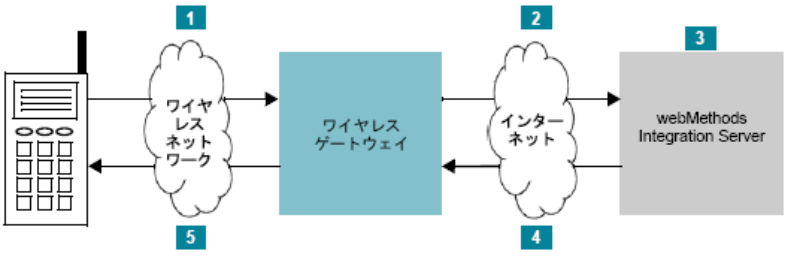
ワイヤレスデバイスから Integration Server にアクセスするには、ワイヤレスデバイスの Web ブラウザで URL を入力します。URL から Integration Server 上のサービスを呼び出したり、Integration Server に保存された WML または HDML ページを要求したりすることができます。

Integration Server とワイヤレスデバイス間の通信方式

Integration Server は、ワイヤレスゲートウェイを使ってワイヤレスデバイスと通信します。ワイヤレスゲートウェイ (WAP ゲートウェイと呼ばれることもある) は、ワイヤレスデバイスからの要求を HTTP 要求に変換します。また、ワイヤレスゲートウェイは、Integration Server からの HTTP 応答もワイヤレスデバイス上の Web ブラウザまたはマイクロブラウザに対応したフォーマットに変換します。

次の図は、Integration Server がインターネットを使用できるワイヤレスデバイスと通信する仕組みを示しています。

Integration Server とワイヤレスデバイス間の通信



ステージ	説明
1	ユーザは携帯電話や PDA (Personal Digital Assistant) などのワイヤレスデバイス上の Web ブラウザを使用して、URL を要求します。URL は、呼び出す対象のサービスを示しているか、要求する WML または HDML ページを示しています。ワイヤレスデバイスは、エンコードされた要求をワイヤレスゲートウェイに送信します。
2	ワイヤレスゲートウェイ (Phone.com の Up.Link Server や Nokia Active Server など) はワイヤレスデバイスからの要求をデコードし、指定した URL に対する HTTP または HTTPS 要求 (URL 内の指定により異なる) を作成して、Integration Server に送信します。
3	Integration Server は、ユーザが URL 内で要求した内容によって、次に示すいずれかの動作をします。 URL に指定されたサービスを実行し、割り当てられた WML または HDML 出力テンプレートにサービス結果を挿入します。 または URL 内で要求された WML または HDML ページを抽出します。
4	Integration Server は、HTTP または HTTPS 応答をワイヤレスゲートウェイに送信します。
5	ワイヤレスゲートウェイは、応答から HTTP または HTTPS ヘッダーを取り除き、HDML または WML コンテンツを含むエンコードされた応答をワイヤレスデバイスに送信します。ワイヤレスデバイス上の Web ブラウザは応答をデコードし、WML または HDML 結果を表示します。

ワイヤレスゲートウェイとワイヤレスプロトコルの詳細については、www.wapforum.org を参照してください。

Integration Server へのワイヤレス接続のための URL 使用

ワイヤレスデバイスを使用して情報にアクセスしたり Integration Server 上のサービスを呼び出したりするには、デバイスの Web ブラウザで URL の入力または選択を行う必要があります。ここでは、URL を使用してサービスを呼び出す方法と、URL を使用して WML または HDML ページを要求する方法について説明します。

注:
ワイヤレスデバイス用の Web ブラウザの中には、ユーザが入力または選択できる URL の長さに制限があるものがあります。ワイヤレスデバイスで使用するために作成する WML または HDML ページが、ブラウザの要件に準拠していることを必ず確認してください。

注:
ユーザが入力する量をできるだけ減らして入力エラーが発生する可能性を低くするには、WML または HDML ページに URL のハイパーリンクを埋め込みます。

URL によるサービスの呼び出し

インターネットを使用できるワイヤレスデバイスからサービスを呼び出すために、URL を使用することができます。URL を要求するには、Web ブラウザに直接 URL を入力するか、HDML または WML ページに埋め込まれている URL のリンクを選択します。いずれの場合も、URL は次のフォーマットになっている必要があります。

項目	説明
1	呼び出す対象のサービスが存在する Integration Server の名前とポート番号を指定します。 **重要:** ワイヤレス接続の場合、サーバ名 (localhost) には登録されたドメイン名を指定する必要があります。つまり、サーバをインターネット経由でアクセスできるようにする必要があります。 **重要:** 多くのワイヤレスゲートウェイでは、ポート 80 をデフォルトの登録済みポート番号として使用します。別のポート番号を使用する場合は、ワイヤレスゲートウェイにサーバ名とポート番号を必ず登録してください(セキュリティ上の理由により、Software AG では 1024 未満のポート番号を使用することをお勧めしていません。詳細については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください。
2	必須キーワード「invoke」を指定して、URL が呼び出し対象のサービスを示していることを Integration Server に通知します。
3	呼び出されるサービスが存在するフォルダを指定します。サブフォルダはピリオドで区切ります。このフィールドは大文字と小文字を区別します。必ず Integration Server のフォルダ名に指定されているとおりに大文字と小文字の組み合わせを使用してください。
4	呼び出されるサービスを特定します。このフィールドは大文字と小文字を区別します。必ず Integration Server のサービス名に指定されているとおりに、大文字と小文字の組み合わせを使用してください。
5	サービスへの入力値を指定します。入力値の前には疑問符 (?) を指定します。疑問符は、入力値の開始を示します。各入力値は、*variable*=*value* の対で表します。*variable* の部分は大文字と小文字を区別します。必ずサービス名に指定されているとおりに、大文字と小文字の組み合わせを使用してください。サービスに複数の入力値が必要な場合は、*variable*=*value* の対をアンパサンド (&) で区切ります。 **注:**

項目	説明
	URL のこの部分は、HTTP GET メソッド使用時にのみ指定します。

URL を使用したサービスの呼び出しの詳細については、『*webMethods Service Development Help*』の「Building a Browser Based Client」を参照してください。

注:
URL を使用してサービスを呼び出す場合は、サービス出力が適切なテンプレートの型 (WML または HDML) に適合していることを確認します。出力テンプレートの作成の詳細については、『*Dynamic Server Pages and Output Templates Developer's Guide*』を参照してください。

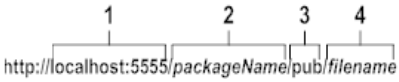
URL による WML または HDML ページの要求

インターネットを使用できるワイヤレスデバイスを使用して、Integration Server に保存された WML または HDML ページを要求することができます。ワイヤレスデバイスの Web ブラウザに URL を入力するか URL へのハイパーリンクを選択することによって、次のディレクトリに保存された WML または HDML ページにアクセスできます。

Integration Server_directory \instances*instance_name*\packages*packageName*\pub

packageName は、WML または HDML ファイルが保存されているパッケージの名前です。

Web ブラウザに入力する URL は、次のフォーマットに従う必要があります。



項目	説明
1	要求する対象のファイルが存在する Integration Server の名前とポート番号を指定します。 **重要:** ワイヤレス接続の場合、サーバ名 (localhost) には登録されたドメイン名を指定する必要があります。つまり、サーバをインターネット経由で外部からアクセスできるようにする必要があります。 **重要:** 多くのワイヤレスゲートウェイでは、ポート 80 をデフォルトの登録済みポート番号として使用します。別のポート番号を使用する場合は、ワイヤレスゲートウェイにサーバ名とポート番号を必ず登録してください(セキュリティ上の理由により、Software AG では 1024 未満のポート番号を使用することをお勧めしていません。詳細については、133 ページの「リモート Integration Server に対するエイリアスの設定」を参照してください)。
2	要求する対象の WML または HDML ファイルがあるパッケージを特定します。

項目	説明
3	pub ディレクトリを指定します。ワイヤレスデバイスに提供される WML および HDML ファイルは、pub ディレクトリに置く必要があります。 **注:** pub ディレクトリは指定しなくてもかまいません。ディレクトリが何も指定されていなければ、Integration Server は自動的に pub ディレクトリ内で要求ファイルを検索します。
4	要求対象のファイルを特定します。

たとえば、Wireless パッケージの pub ディレクトリの hello.wml ファイルにアクセスするには、次に示す URL を使用します。

http://localhost:5555/Wireless/pub/hello.wml

または

http://localhost:5555/Wireless/hello.wml

WML と HDML のサンプル

webMethods Integration Server には、ワイヤレスデバイスが Integration Server と通信する方法を示したサンプルのサービス、WML ファイル、HDML ファイルおよび出力テンプレートがあります。これらのファイルは、WmSamples パッケージ内の sample.wireless フォルダに保管されています。サンプルでは、ワイヤレスデバイスでの製品の注文、注文履歴の表示、Integration Server の統計情報の取得および現在の日付と時刻の要求を行うために作成するサービスの例が提供されています。

サンプルの使用の詳細については、次のファイルを参照してください。

WmSamples\pub\WAPDemo.htm

WmSamples パッケージは、Empower Product Support websiteにある Knowledge Base の認定済みサンプルのセクションにあります。

H Integration Server セッションとサーバログのセッション ID のマスク

Integration Server で、権限のある管理者ユーザは、セッションとサーバログのセッション ID をセキュリティで保護できます。セッション ID をセキュリティで保護するオプションを有効にすると、Integration Server はアスタリスク (*) 文字を使用して、ログのセッション ID 文字列をマスクします。したがって、ログエントリにアクセスするユーザが実際のセッション ID を表示することはできません。これによって正規ユーザのアクティブセッションへの悪意のあるアクセスを防ぎ、Integration Server セッションのセキュリティが向上します。

注:
Integration Server でセッションログエントリが監査ログに書き込まれるとき、セッション ID はマスクされません。監査ログの設定の詳細については、『*webMethods Audit Logging Guide*』を参照してください。

権限のある管理者ユーザは、Integration Server インストールディレクトリの maskSessionID.properties ファイルを編集して、セッション ID をセキュリティで保護できます。

重要:

- Administrators ACL に属し、maskSessionID.properties ファイルへの書き込みアクセス権があるユーザのみ、その内容を編集できます。他のすべての管理者ユーザには、このプロパティファイルへの読み取りアクセス権のみあります。
- 権限のある管理者ユーザは maskSessionID.properties ファイルの内容を手動で編集する必要があります。

セッション ID をマスクするには

1. *Integration Server_directory* \instances*instance_name*\config\security\session\ ディレクトリにある maskSessionID.properties ファイルに移動します。

2. maskSessionID.properties を開き、`maskSessionID = true` を設定します。

3. 変更を保存して、Integration Server を再起動します。

   注:
   maskSessionID.properties ファイルの内容を更新する場合、常に Integration Server を再起動する必要があります。

I サーバログ機能

- サーバログ機能について 1320
- Integration Server 1320
- WmJSONAPI パッケージ 1326
- WmParquet パッケージ 1326
- WmXSLT パッケージ 1326
- フラットファイル 1327

サーバログ機能について

サーバログにより、Integration Server の主要なサブシステム (機能と呼ばれます) の運用情報およびエラー情報を得ることができます。

Integration Server がサーバログに追加する機能を選択します。サーバログの詳細については、235 ページの「サーバログの設定」を参照してください。

Integration Server

次の表に、サーバが情報をログに記録する Integration Server 機能のリストを示します。

機能		情報およびエラーの対象
0000	一般デバッグ	リストにある他の機能に含まれない情報のタイプ。
0001	License Manager	Integration Server の License Manager。
0002	LDAP 接続	LDAP サーバの接続。
0003	Database Connection Manager	Database 接続。
0004	JDBC 接続	JDBC 接続プール。
0006	サーバSSL インタフェース	Integration Server SSL 接続。
0007	承認	Integration Server での承認。
0008	NIS 接続	LDAP サービスと同様のネットワークリソース情報を格納するネットワーク情報サービスサーバ。
0009	認証あり	認証の処理および管理。
0011	プロキシ	プロキシサーバ。
0012	認証	Integration Server に対する認証。
0013	コンテンツハンドラ	コンテンツハンドラ。
0014	サーバ	Integration Server の起動とシャットダウン、スレッドプーリングおよび Integration Server のその他の低レベル処理。0001 License Manager 機能に含まれない、License Manager 関連の Error Manager を含みます。
0015	Service Manager	Integration Server で実行されるフローサービスおよびコーディングされたサービス。
0016	サービスキャッシュ	サービスキャッシュ
0017	Ehcache	Ehcache の設定。
0018	ユーティリティクラス	Integration Server で使用するユーティリティクラス。

機能		情報およびエラーの対象
0019	OpenID Connect	OpenID Connect。
0020	サーキットブレーカー	サーバに設定されたサーキットブレーカー。
0021	Integration Cloud	Integration Cloud アクティビティ。
0022	コードの生成	Designer を使用して生成された、別のサービスまたはクライアントからサービスを実装またはサービスを呼び出すためのコード。
0024	User Manager	ユーザ管理。
0025	サーバの初期化	サーバの初期化。
0026	サービス	Integration Server で実行されるフローサービスおよびコーディングされたサービス (Java など)。
0027	サーバの設定	HSM ベースのキーストアおよび nCipher。
0028	パッケージ	パッケージのロード、アンロードおよびその他の処理。
0030	Admin API	Integration Server Administrator API。
0033	Cluster Manager	クラスタ化された Integration Server。
0036	クライアントコンテキスト	クライアントのトランザクション。
0037	保証付きデリバーコンテキスト	保証付きデリバーのトランザクション。
0038	HTTP ヘッダー	HTTP のヘッダー。
0039	HTTP 要求	Integration Server への HTTP 要求。
0040	HTTP 応答	Integration Server からの HTTP 要求。
0041	HTTP Cookie	HTTP の Cookie。
0042	XML パーサ	XML パーサ。
0043	XML 解析ストリーム	ストリームから XML データを読み取る XML パーサ。
0045	Statistics Data Collector	ダッシュボードで使用されるデータの収集、インデックス作成、および保存を行う Statistics Data Collector。
0046	HTTP リスナー	HTTP のリスナー。
0047	HTTPS リスナー	HTTPS のリスナー。
0048	各種サーバエラーメッセージ	0014 サーバ機能に含まれない各種エラー。
0049	フロー操作	フロー操作。

機能		情報およびエラーの対象
0050	フローマップ	フローサービス内の MAP ステップおよび INVOKE ステップ内のパイプラインデータのマッピング。
0051	Port Manager	Integration Server のポート。
0053	HTTP ディスパッチ	HTTP 要求の処理。
0054	HTTP ドキュメントハンドラ	HTTP 応答に登録されたドキュメントハンドラ。
0055	Java サービス	Java サービス。
0056	メーラー	Integration Server が使用するメーラー。
0057	Deployer	Deployer
0059	サービススレッド	サービススレッド。
0061	リモートサーバ	リモートサーバ。
0062	コアサービス	XML および XQL の処理。
0063	Transaction Job Manager	トランザクションマネージャ。
0064	ネットワークサービス	ネットワークのサービス。
0068	電子メールリスナー	電子メールのリスナー。
0070	リスナー	Integration Server に定義されているリスナー。
0071	FTP リスナー	FTP のリスナー。
0072	レポータ	DSP 処理。
0075	WmDB	WmDB パッケージ。
0076	コーダー	ファイルに保存したり、ネットワークでしたり送信するために、Java オブジェクトをバイトにシリアライズするとき、またはその逆を行うときに Integration Server によって使用される、IDataBinCoder などのコーダー。
0077	com.wm.util	com.wm.util コーダー。
0079	Enterprise Gateway	webMethods Enterprise Gateway Server アクティビティ。
0080	非同期接続 API	ネットワークソケット。
0081	ネームスペース	ネームスペースの設定。
0082	XML スキーマ	XML スキーマ。
0085	WmRoot パッケージ	WmRoot パッケージ。
0087	WmWin32 パッケージ	WmWin32 パッケージ。

機能		情報およびエラーの対象
0088	SOAP	SOAP メッセージ。
0090	pub フローサービス	認証処理、pub.flow サービスおよび WSDL 処理。
0091	WmRoot 管理サービス	WmRoot 管理のサービス。
0094	リポジトリ V4	リポジトリバージョン 4。
0095	Audit Log Manager	監査ログマネージャ。
0096	JDBC Connection Manager	JDBC 接続プール。
0097	Broker ドキュメントタイプシンクロナイザ	パブリッシュ可能なドキュメントタイプを Broker の Broker ドキュメントタイプと同期します。
0098	ディスパッチャ	トリガー、 Broker上の JMS クライアントおよび Broker クライアント。
0099	Broker 転送レイヤ	Broker 接続。
0100	Web コンテナ	Tomcat JSP エンジンおよび Tomcat JSP エンジンをホストするコンポーネント。
0101	プロセスランタイム	WmPRT パッケージ。
0105	相互参照とラッチ	クラスタに属する Integration Server における相互参照およびラッチ。
0106	Join Manager	トリガー処理で使用される結合。
0109	サーバの統計情報	Integration Server のセッションおよびスレッドアクティビティに関する統計情報。
0114	Adapter Runtime	Adapter Runtime 機能。
0115	Adapter Runtime (リスナー)	アダプタ接続を使用してアダプタリソースに接続するアダプタリスナー。
0116	Adapter Runtime (通知)	アダプタ通知 (ポーリング通知およびリスナー通知)。
0117	Adapter Runtime (アダプタサービス)	アダプタリソースに対して実行されるアダプタの操作を定義するアダプタサービス。
0118	Adapter Runtime (接続)	アダプタ通知およびアダプタリスナーがアダプタリソースへの接続に使用するパラメータを含むアダプタ接続。
0119	Monitor	webMethods Monitor
0120	監視 (データベースレイヤ)	webMethods Monitor データベースレイヤ。
0121	Adapter Runtime (SCC Transaction Manager)	Adapter Runtime (JCA System Contract Component Transaction Manager)。

機能		情報およびエラーの対象
0123	Basis FSData	リポジトリ、一時ストアおよびフラットファイルのロギングなど、他の Integration Server コンポーネントの基盤となる記憶領域システムとして使用されるファイルシステム。これらのコンポーネントで発生したエラーの詳細情報を提供します。
0125	監視制御メカニズム	リソース (メモリ、スレッドまたはディスク) の枯渇を防止するための Integration Server のリソース。現在は、パブリッシュ/サブスクライブおよびスレッドプールに割り当てられたリソースのみを制御します。
0126	Adapter Runtime (SCC Connection Manager)	Adapter Runtime (JCA System Contract Component Connection Manager)。
0127	Basis 一時ストア	一時ストア (ロギングに使用するストアなど)。
0128	重複抑制処理プロセッサ	webMethods messaging trigger または JMS トリガー用の重複抑制処理プロセッサ。
0129	.NET パッケージ	.NET のサービス。
0130	Mainframe パッケージ	Mainframe パッケージ。
0131	診断ポート	Integration Server 診断ポート。
0132	バージョン管理システム	Integration Server のバージョン管理システム統合機能。
0133	DOM 実装	XML ノードを DOM ノードとして作成した場合に使用、または DOM ノードとして使用します。
0134	JMS サブシステム	JMS サブシステム。
0135	JNDI クライアント設定	JNDI プロバイダ管理、接続、およびプロバイダから返された JNDI オブジェクトを含む JNDI クライアント設定。
0137	ユーザタスクスケジューラ	ユーザが作成したスケジュール済みタスク。
0138	Asset Publisher	Asset Publisher パッケージ。
0139	キーストア	キーストア。
0140	Designer	Software AG Designer
0141	Web サービススタック	Web サービススタック。
0142	スレッド強制終了	サービスが無応答状態になった場合にサービススレッドをキャンセルまたは強制終了できるスレッド強制終了機能。
0144	XML セキュリティサービス	pub.security.xml のサービス。

機能		情報およびエラーの対象
0147	SFTP クライアント	SFTP クライアントとして機能する Integration Server。
0149	Enterprise Gateway ルール	webMethods Enterprise Gateway ルール。
0150	SPM	Integration Server 内の Software AG Platform Manager サブシステム。
0151	監視 (CMP)	コンテンツ監視プラットフォーム。
0152	CORS	CORS (クロスオリジンリソース共有) のサポート。
0153	ディスパッチャ (WM メッセージング)	Universal Messagingからメッセージまたはドキュメントを受信するトリガー。
0154	Protocol Buffer のエンコーディング (Universal Messaging)	プロトコルバッファのエンコードとデコード、およびパブリッシュ可能なドキュメントタイプと Universal Messaging の関連するチャネルとの同期。
0155	ホット展開	Integration Server アセットのホット展開。
0156	イベントルーティング	WmPublic の pub.event.routing サービス。
0157	アセット	WmPublic の pub.assets サービス。
0158	OData	OData サービス。
0159	Digital Event Services	Digital Event Services を使用したイベントの作成、送信、受信。
0160	検索とリファクタリング	Integration Server アセットの検索とリファクタリング。
0161	WebSocket	WebSocket エンドポイントプロビジョニングおよびコールバック呼び出し。
0162	Ehcache IData	Ehcache はリポジトリ、一時ストアおよびフラットファイルのロギングなど、他の Integration Server コンポーネントの基盤となる記憶領域システムとして使用されます。これらのコンポーネントで発生したエラーの詳細情報を提供します。
0163	JSON Web Token	JSON Web Token の認証と設定。
0164	JSON 妥当性検査	スキーマに対する JSON コンテンツの妥当性検査。
0165	WebSocket リスナー	WebSocket リスナーのプロビジョニングとライフサイクル。
0166	REST	REST リソースおよび REST API 記述子。
0167	HTTP インターセプタ	HTTP インターセプタフレームワークの初期化と操作。

機能		情報およびエラーの対象
0168	メッセージング (拡張ロギング)	拡張ロギングの設定時のメッセージの送受信。
0169	アカウントロック	ユーザアカウントのロック。
0170	JSON スキーマ	JSON スキーマ。
0171	パッケージの自動展開	パッケージの自動展開のインストール、アクティブ化およびその他の処理。
0172	GraphQL	GraphQL 操作。
0173	共通メッセージング	MQTT メッセージング。
0174	アラートフレームワーク	アラートフレームワーク。
0175	アラート通知者	アラートの通知者。
0176	メータリング	webMethods Metering Agent。
0177	GRPC フレームワーク	gRPC 操作。
0178	スキャフォールディング	スキャフォールディング、エージェントマニフェストの生成、およびクラウド同期用のアプリケーション。

WmJSONAPI パッケージ

次の表に、サーバが情報をログに記録する WmJSONAPI パッケージ機能の一覧を示します。

機能		情報およびエラーの対象
0801	CORE	webMethods JSONAPI のサポートに関連する情報およびエラー

WmParquet パッケージ

次の表に、サーバが情報をログに記録する WmParquet パッケージ機能の一覧を示します。

機能		情報およびエラーの対象
0001	サービス	webMethods Parquet のサポートパブリックサービスに関連する情報およびエラー。

WmXSLT パッケージ

次の表に、サーバが情報をログに記録する WmXSLT パッケージ機能のリストを示します。

機能		情報およびエラーの対象
0001	SAX	SAX 解析の関連情報。
0002	JAXP	JAXP 関連の情報。XML パーサおよび XSLT エンジンのメッセージが含まれます。
0003	サービス	XSLT サービスのパブリックサービス情報。
0004	Admin サービス	XSLT サービスの非パブリック管理サービス情報。

フラットファイル

次の表に、サーバが情報をログに記録するフラットファイル機能のリストを示します。

機能		情報およびエラーの対象
0000	フラットファイルロガー	フラットファイルの処理。
0001	フラットファイルリスナー	フラットファイルのポーリングリスナー。
0002	フラットファイルパーサ	フラットファイルのパーサ。