Administering webMethods CloudStreams

Version 10.7

October 2020

WEBMETHODS

This document applies to webMethods CloudStreams 10.7 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: WST-CLOUDSTREAMS-USERS-GUIDE-107-20201015

Table of Contents

About this Guide

CloudStreams provides a scalable approach for the development and governance of integration flows between Software as a Service (SaaS) applications and on-premises applications.

CloudStreams provides predefined, configurable CloudStreams connectors that enable you to connect to popular SaaS cloud applications. Using the CloudStreams plug-ins and administrative options, you configure these connectors to govern transactions, log payloads and monitor run-time performance. You can publish and view run-time performance metrics and events using the CloudStreams Analytics dashboard.

Additionally, you can use the CloudStreams framework to create connectors to other SaaS applications.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at https://empower.softwareag.com/.

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

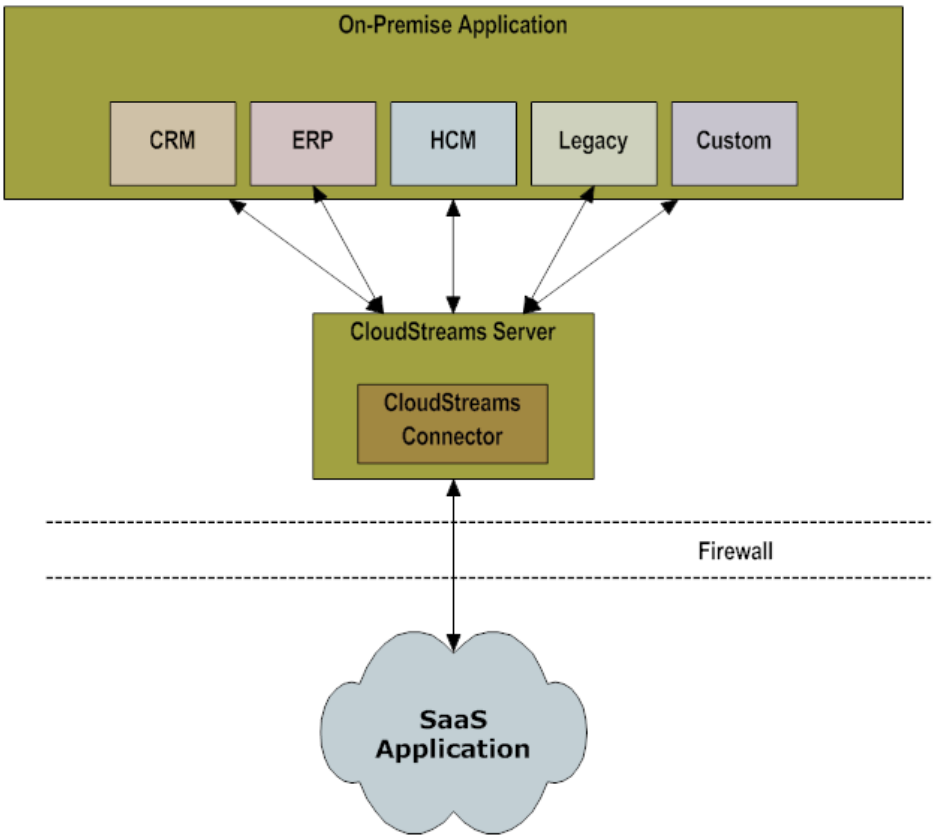
Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Overview

About CloudStreams

CloudStreams provides a scalable approach for the development and governance of integration flows between “Software as a Service” (SaaS) applications and on-premises applications.



You use CloudStreams to:

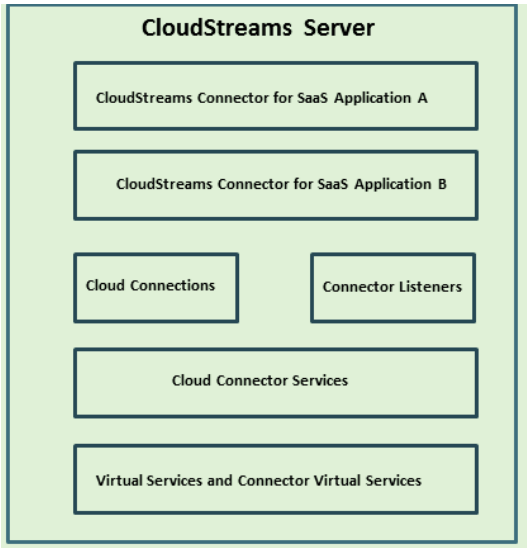
- Invoke SOAP/REST APIs using CloudStreams cloud connector services
- Listen and process streaming API events using CloudStreams connector listeners
- Manage the security of service requests
- Perform user-defined processing instructions (optional)
- Log transactions
- Monitor run-time performance and events

Additionally, you can use the CloudStreams framework to create custom cloud connectors for other SaaS applications.

CloudStreams Components

CloudStreams consists of the following components:

- **CloudStreams Server**, which uses Integration Server or webMethods Microservices Runtime as the runtime. CloudStreams server runs as a package on top of Integration Server or webMethods Microservices Runtime. CloudStreams server supports the run-time behavior for the following components:



Note:
Additionally, the CloudStreams Server provides a dashboard that can be imported and viewed on the Software AG MashZone NextGen Server. The dashboard provides a visual view of the run-time performance metrics and run-time events of CloudStreams providers, cloud connector services, and consumers.

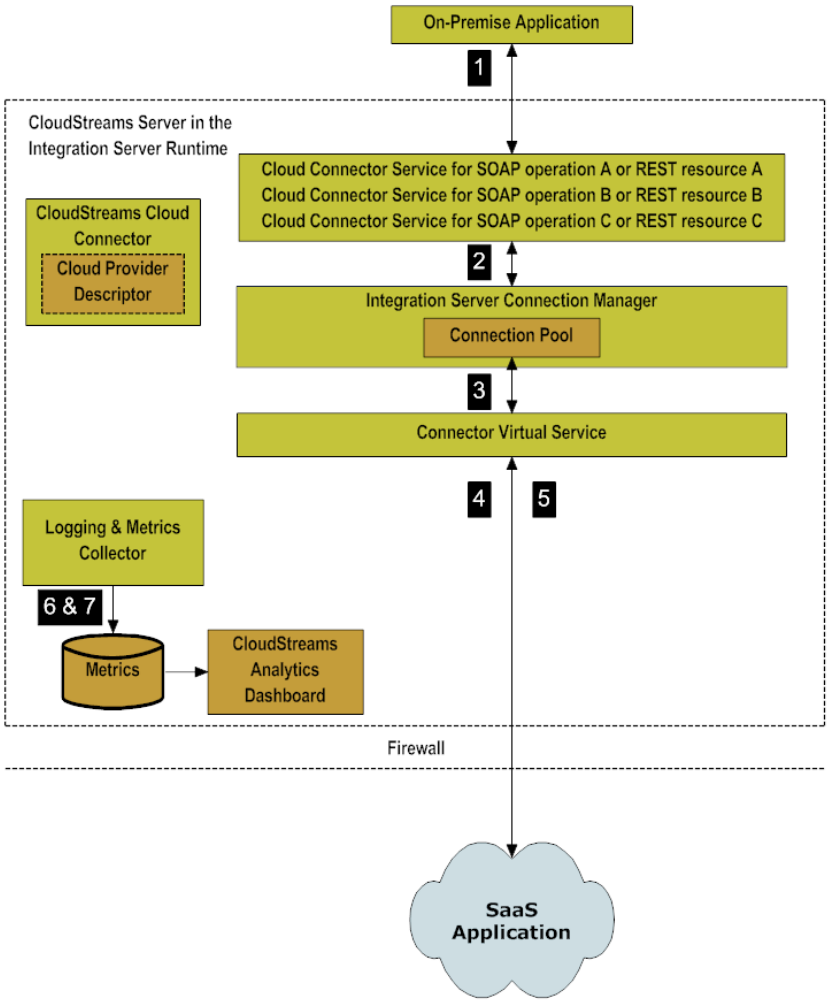
- **Plug-ins provided by Software AG Designer**, which you use as the design-time tool for CloudStreams:
  - You use the **Service Development** plug-in to create cloud connector services, which perform SOAP operations or access REST resources. You also use the **Service Development** plug-in to create and update connector listeners to listen to and process streaming events.
  - You use the **CloudStreams Development** plug-in to create providers, connectors, virtual services, connector virtual services, and their policies. These services send the requests from the on-premises application to the native service in the SaaS application, and vice versa.

Note:
You can deploy CloudStreams user-created assets that reside on source webMethods repositories to target webMethods run-time components (runtimes) using webMethods Deployer.

Note:
Currently, webMethods Mediator and CloudStreams products are not compatible and they must be installed on separate webMethods Integration Server instances.

SOAP/REST Outbound Architecture

In the outbound run-time scenario, an on-premises application invokes a SaaS application through CloudStreams as follows:

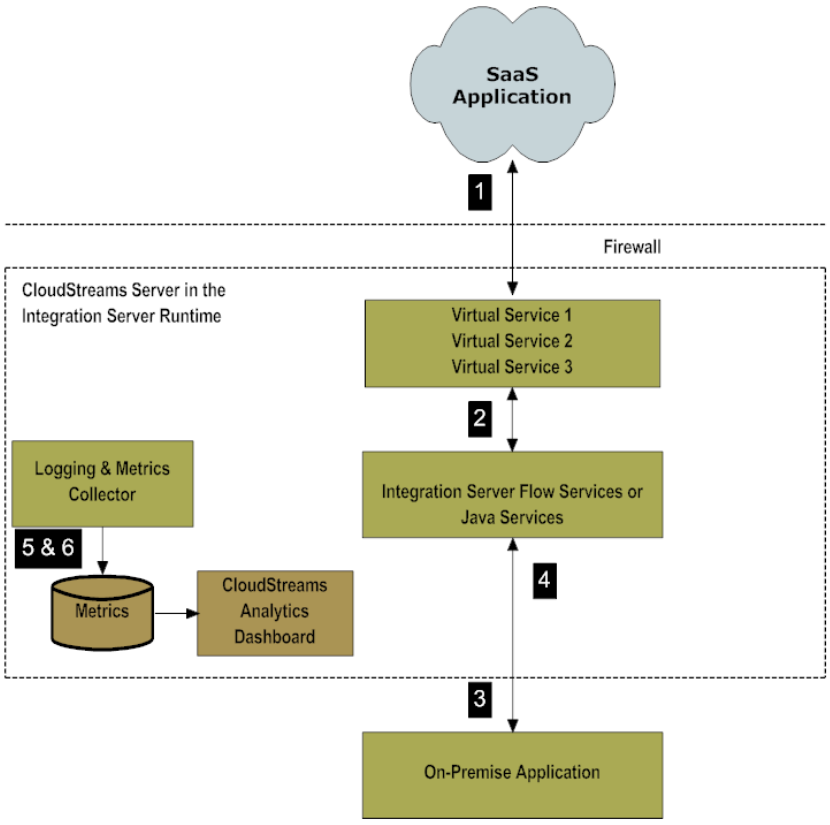


Step	Description
1	The on-premises application invokes a cloud connector service that you defined to perform the desired SOAP operation (such as querying an account) or to access the desired REST resource.
2	The cloud connector service obtains a connection from the Integration Server connection pool.
3	The cloud connector service then invokes a connector virtual service.
4	The cloud connector service sends the request to the native service in the SaaS application.
5	The SaaS application processes the request for the native service and sends a response to the connector virtual service.

Step	Description
6	The CloudStreams server collects run-time metrics and events, and publishes them to the CloudStreams Analytics facility, which is powered by Software AG MashZone NextGen Server. You can use the CloudStreams Analytics dashboard to monitor the metrics.
7	The connector virtual service logs the request/response transaction and its payload.

SOAP/REST Inbound Architecture

In the inbound run-time scenario, a SaaS application invokes an on-premises application through CloudStreams as follows:

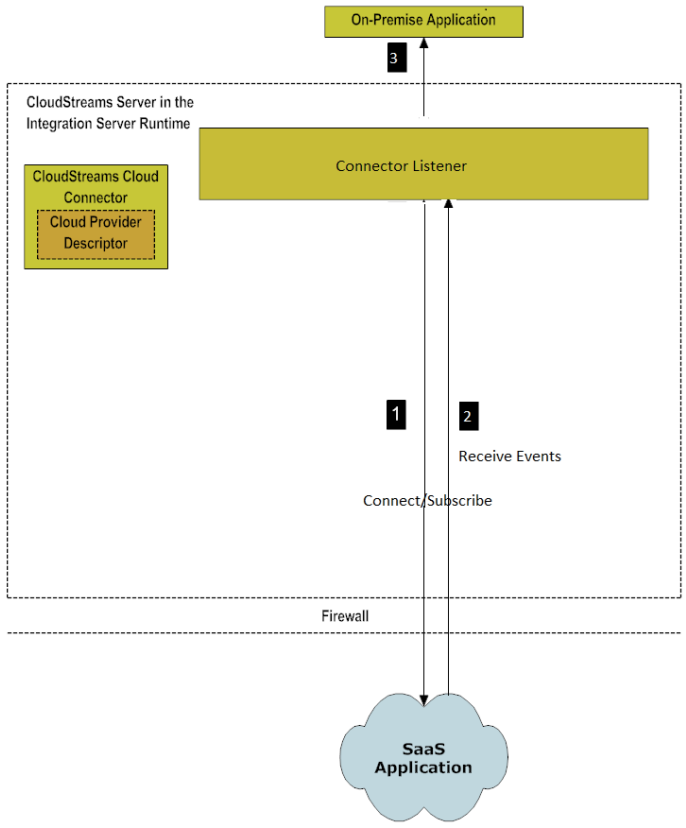


Step	Description
1	A SaaS application invokes a virtual service that has been deployed to CloudStreams. The virtual service's policy performs security checks and any other user-defined processing before sending the request to the on-premises application.
2	The virtual service invokes a service endpoint (an Integration Server flow service or a Java service) in the on-premises application.
3	The endpoint service then invokes the on-premises application service(s).

Step	Description
4	The on-premises application processes the request against the native service and sends a response back to the SaaS application.
5	The CloudStreams server collects run-time metrics and events, and publishes them to the CloudStreams Analytics facility, which is powered by Software AG MashZone NextGen Server. You can use the CloudStreams Analytics dashboard to monitor the metrics.
6	The virtual service logs the request/response transaction and its payload.

Streaming Architecture

1. The connector listener establishes a connection with the streaming endpoint.
2. The subscribed events are received.
3. The received events are posted to the on-premise application for processing.



Note:
See " Cloud Connections, Services, and Connector Listeners" on page 197 for more information on CloudStreams providers, connectors, cloud connections, cloud connector services, connector listeners, and how to manage cloud connectors and cloud connector packages.

Virtual Services and Connector Virtual Services

A *virtual service* is a service that runs on a CloudStreams server and acts as the consumer-facing proxy for a native service that runs in a SaaS application or in an on-premises application. Service requests go to a virtual service for processing rather than directly to the service provider. You should create a virtual service for each SaaS service you want to expose to consumers.

There are two types of virtual services: *virtual services* and *connector virtual services*.

- In the inbound processing scenario, when a SaaS application sends a service request, a virtual service's policy performs security checks and any other user-defined processing before sending the request to the on-premises application.
- In the outbound processing scenario, when an on-premises application sends a service request, a special kind of virtual service is used, called a connector virtual service. A *connector virtual service* handles the SaaS provider's responses and logs the requests/responses and their payloads.

  CloudStreams provides a default connector virtual service for each metadata handler (one for the SOAP handler and one for the REST handler). You cannot modify these default services, which are located in the WmCloudStreams package. Alternatively, you can create additional connector virtual services with custom policies.

You create both types of virtual services using the CloudStreams Development plug-in provided in Designer, and then deploy them to a CloudStreams server.

Both types of virtual services can have user-defined *policies*, which provide governance capabilities for the services. Policies are discussed later in this overview.

The Processing Steps of Virtual Services

When you create a virtual service, you can configure the following processing steps:

- The service's **In Sequence step**, which you configure to manipulate the request messages. This step can include the following sub-steps:
  - The **Entry Step** (provided by default), which specifies the protocol (HTTP or HTTPS) of the requests that the virtual service will accept. For REST-based virtual services, you also specify the REST resource that the service will access and the HTTP methods (GET, POST, PUT and DELETE) that the virtual service should be allowed to perform on the REST resource.
  - The **Transform step** (optional), which performs an XSLT message transformation on the request message before the virtual service submits it to the native service.
  - The **Invoke IS Service step** (optional), which preprocesses the request message before the virtual service submits it to the native service.
  - The **Routing Rule step** (provided by default), which specifies how the virtual service will route the requests to the native service endpoint(s). For HTTP or HTTPS requests, you have four routing choices:

    - "Straight Through" routing (to route requests directly to the native service endpoint).
    - "Context-Based" routing (to route specific types of messages to specific endpoints according to context-based routing rules).
    - "Content-Based" routing (to route specific types of messages to specific endpoints based on specific values that appear in the request message).
    - "Load Balancing" routing (to distribute requests among multiple endpoints).
- The service's **Out Sequence step**, which you configure to manipulate the response messages. This step can include the following sub-steps:
    - The **Transform step** (optional), which specifies how the response message from the native service provider is to be transformed before the virtual service returns it to the consuming application.
    - The **Invoke IS Service step** (optional), which preprocesses the response message before the virtual service returns it to the consuming application.
- The service's **Error Sequence step** (provided by default). CloudStreams returns a default fault response to the consuming application, which you can customize with context variables. This fault response is used for faults returned by the native service provider as well as faults returned by internal CloudStreams exceptions, such as policy violation errors, connection timeouts, etc. You can choose whether or not to send the native service provider's service fault content, or just send the message. In addition, you can invoke IS services to pre-process or post-process the error messages.

The Processing Steps of Connector Virtual Services

A connector virtual service can contain similar processing steps as a virtual service.

Policies for Virtual Services and Connector Virtual Services

Policies provide run-time governance capabilities for virtual services and connector virtual services.

Policies for Virtual Services

You should create policies that apply to one or more virtual services. A *policy* is a sequence of actions that is carried out by CloudStreams when a consumer requests a particular service through CloudStreams. A policy for a virtual service can include the following kinds of actions:

- **WS-SecurityPolicy 1.2 actions:**

  There are two kinds of WS-SecurityPolicy 1.2 actions:

    - Authentication actions, to verify that the requests for virtual services contain a specified WS-SecurityPolicy element. You can authenticate consumers by their WSS X.509 certificates, WSS username tokens, or WSS SAML tokens.
    - XML security actions, to provide confidentiality (through encryption) and integrity (through signatures) for request and response messages.

- **Monitoring actions:**
  - The “Monitor Service Performance” action, which monitors a user-specified set of run-time performance conditions for a virtual service, and sends alerts to a specified destination when these conditions are violated
  - The “Monitor Service Level Agreement” action, which provides the same functionality as “Monitor Service Performance”, but this action is different because it enables you to monitor a virtual service's run-time performance for particular consumers. You configure this action to define a *Service Level Agreement* (SLA), which is set of conditions that defines the level of performance that a specified consumer should expect from a service.
  - The “Throttling Traffic Optimization” action, which limits the number of service invocations allowed during a specified time interval.
- **Additional actions:**
  - “Identify Consumer”, which you use in conjunction with an authentication action. Alternatively, this action can be used alone to identify consumers only by host name or IP address.
  - “Require HTTP Basic Authentication”, which uses HTTP Basic authentication to verify the consumer's authentication credentials contained in the request's Authorization header against the Integration Server's user account. This action supports WS-SecurityPolicy 1.2.
  - “Authorize User”, which authorizes consumers against a list of users and/or a list of groups registered in the Integration Server. You use this action in conjunction with an authentication action.
  - “Log Invocation”, which logs request/response payloads. See “Log Invocation” on page 171 for General Data Protection Regulation (GDPR) related information.
  - “Validate Schema”, which validates all XML request and/or response messages against an XML schema referenced in the WSDL.

Policies for Connector Virtual Services

Each default connector virtual service has a default policy, which logs the requests/responses and their payloads to a database. You cannot modify the default connector virtual services or their policies. However, you can create additional connector virtual services with custom policies.

If you create a connector virtual service with a custom policy, you can include the actions from the “Monitoring” and “Additional” categories (except for “Validate Schema”); the “WS-SecurityPolicy 1.2” actions are not needed. For example, you might want to create a custom policy that monitors run-time performance, customizes how the service invocations are logged, or optimizes server traffic.

The CloudStreams Analytics Dashboard

CloudStreams can publish key performance indicator (KPI) metrics and run-time events to the CloudStreams Analytics dashboard. The dashboard enables you to view and analyze the KPI

metrics and run-time events of your CloudStreams providers, cloud connector services, and consumers.

The CloudStreams Analytics dashboard is powered by Software AG MashZone NextGen Server. The dashboard tabs are backed by data feeds that communicate with the database using the JDBC connection pool of Integration Server to publish data at regular intervals.

Note:
The datafeed of Software AG MashZone NextGen dashboard fetches data through a REST service hosted on Integration Server to retrieve metrics and events from the database and send it back to Software AG MashZone NextGen.

Note:
The CloudStreams Analytics dashboard has been updated and migrated from the legacy Software AG MashZone to Software AG MashZone NextGen.

The dashboard displays the following analytic views:

View	Can be filtered by...
Performance and availability for providers	
API usage by provider(s), cloud connector service(s), and consumer(s)	Provider and time period
Performance trends for provider(s) and cloud connector service(s)	Provider and time period
Error trends for provider(s) and cloud connector service(s)	Provider and time period
Invocation details for cloud connector service(s)	Provider, cloud connector service, and time period

The types of KPI metrics and run-time events in the views are listed below.

The Run-Time Events

The types of run-time events that can be published are as follows:

Event Type	Description
Lifecycle	A Lifecyle event occurs each time a CloudStreams provider is started or shut down.
Transaction	A Transaction event occurs each time a cloud connector service is invoked (successfully or unsuccessfully).
Error	An Error event occurs each time an invocation of a cloud connector service results in an error.

Event Type	Description
Policy Violation	A Policy Violation event occurs each time an invocation of a cloud connector service violates a policy that was set for its associated virtual service.
Monitoring	CloudStreams publishes key performance indicator (KPI) metrics, such as the average response time, fault count, and availability of all providers and connector services.

The Key Performance Indicator (KPI) Metrics

The types of KPI metrics that can be published are as follows:

Metric	Description
Availability	Indicates whether the provider was available to the specified consumers in the current interval. For information about intervals, see "Setting the Database Options for Publishing Performance Metrics and Events" on page 32. A value of 100 indicates that the connector or service was always available. If invocations fail due to policy violations, this parameter could still be as high as 100.
Average Response Time	The average amount of time it took the cloud connector service to complete all invocations in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller.
Fault Count	The number of failed invocations for a cloud connector service in the current interval.
Success Count	The number of successful invocations for a cloud connector service in the current interval.
Total Request Count	The total number of requests made by a connector, cloud connector service, or consumer in the current interval.

To enable CloudStreams to publish KPI metrics and events to the CloudStreams Analytics dashboard, you must set the options in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

Note:
The metrics do not include metrics for failed invocations unless you set the `pg.PgMetricsFormatter.includeFaults` parameter to true in *`IntegrationServer_directory`*`\instances\`*`instance_name`*`\packages\WmCloudStreams\config\resources\wst-config.properties`. For more information, see "Advanced Settings" on page 281.

Getting Started with CloudStreams

Step	For procedures, see...
1. Using the Software AG Installer, install the CloudStreams Server and the Software AG MashZone NextGen server. The Software AG MashZone NextGen Server is required only if you plan to use CloudStreams Analytics. Also, download the CloudStreams Provider (such as the Salesforce.com Provider) from the Software AG TECHcommunity website. **Note:** From the 9.7 release, CloudStreams providers and provider user guides are available for download from the TECHcommunity website. The provider installation guide is available in the CloudStreams section of the Software AG Documentation website.	■ The document *Installing Software AG Products*. ■ "CloudStreams Analytics" on page 247 ■ The documentation specific to your CloudStreams provider (for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*).
2. Create a cloud connection and configure the session management parameters, using the Integration Server Administrator.	The documentation specific to your CloudStreams provider.
3. Create a cloud connector service for each desired operation defined in the cloud provider's WSDL, or for each desired REST resource, using the Service Development plug-in available in Software AG Designer.	The documentation specific to your CloudStreams provider.
4. Create a connector listener for each subscription endpoint, using the Service Development plug-in available in Software AG Designer.	"Working with Connector Listeners" on page 200
5. Set the CloudStreams configuration options to specify things such as the keystore, truststore and ports to use, the database to use for publishing performance metrics and events, the consumer applications that can access CloudStreams, and more.	"CloudStreams Configuration Options" on page 25
6. Define a CloudStreams Server target, which specifies an instance of a CloudStreams server to which you will deploy your virtual services and any optional custom connector virtual services. You do this using Software AG Designer. You can define one or more server targets.	"Virtual Services" on page 55

Step	For procedures, see...
7. Create a CloudStreams Governance project in which you will create the virtual services, connector virtual services, and their policies. You create projects in your local file system, using the CloudStreams Development plug-in available in Software AG Designer. You can create one or more projects.	"Virtual Services" on page 55
8. In the CloudStreams Governance project, create virtual services and connector virtual services. You do this using the CloudStreams Development plug-in available in Software AG Designer. You will: ■ Create virtual services to handle the inbound requests. ■ Optionally override the default connector virtual services with custom ones to handle outbound requests. CloudStreams provides a default connector virtual service for each metadata handler (which you cannot modify), but you can create additional connector virtual services with custom policies, if desired.	"Virtual Services" on page 55
9. In the CloudStreams Governance project, create policies for the virtual services using the CloudStreams Development plug-in available in Software AG Designer. ■ For the virtual services, you create policies that include WS-SecurityPolicy actions (and other actions) provided by CloudStreams. ■ For the connector virtual services, CloudStreams provides a default policy that logs all outbound transactions. You cannot modify these policies, but you can create additional connector virtual services with custom policies that include the monitoring, logging, and schema validation actions.	"Policies" on page 159
10. Deploy the virtual services and any custom connector virtual services to a package in a CloudStreams server target.	"Deploying Virtual Services and Connector Virtual Services" on page 191
11. Configure CloudStreams Analytics to monitor performance metrics and run-time events.	"CloudStreams Analytics" on page 247

Note:

See the *webMethods CloudStreams FAQ and Troubleshooting* document available in the CloudStreams section of the Software AG Documentation website for FAQs and Troubleshooting tips.

2 CloudStreams Configuration Options

Overview

Set the CloudStreams configuration options to specify things such as the keystore, truststore and ports to use, the database to use for publishing run-time metrics and events, the consumer applications that are allowed to access CloudStreams, and more.

The Administration Options

In Integration Server Administrator, you can set the following options under **Solutions > CloudStreams > Administration**:

Setting the General Options

Use these options to manage the inbound service requests sent by consumers to CloudStreams. To manage the *outbound* requests that CloudStreams sends to the cloud provider, set the options for your cloud connections in your CloudStreams cloud connector. For details, see the section *Creating Cloud Connections* in the documentation for your CloudStreams cloud connector.

To set the CloudStreams > Administration > General options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > General**.

2. Click **Edit**, complete the following fields and click **Save**.

Option	Description
Target Name	The CloudStreams target name. This name is included in the events that are sent to the database for metrics reporting purposes.

Option	Description

HTTP Load Balancer URL	The primary HTTP load balancer URL and port number to use.
	
	For the URL, you can specify either the IP address or host name of the load balancer with the port number, as follows:
	
	`http://IP-address:portnumber`
	
	or
	
	`http://hostname:portnumber`
	
	If specified, all virtual services hosted in CloudStreams will use this value.

HTTPS Load Balancer URL	The primary HTTPS load balancer URL and port number to use.
	
	For the URL, you can specify either the IP address or host name of the load balancer with the port number, as follows:
	
	`https://IP-address:portnumber`
	
	or
	
	`https://hostname:portnumber`
	
	If specified, all virtual services hosted in CloudStreams (and exposed on HTTPS) will use this value.

Keystore Name	The Integration Server keystore that CloudStreams should use. This field lists all available Integration Server keystores. If there are no configured Integration Server keystores, the list will be empty. To configure Integration Server keystores, see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*.
Signing Alias	The signing alias. This alias is the value that is used to sign the outgoing response from CloudStreams to the original consumer. It is auto-populated based on the keystore selected from the **IS Keystore Name** above. This field will list all the aliases available in the chosen keystore. If there are no configured keystores, this field will be empty.
Truststore Name	Specify the same truststore that is specified in Integration Server (as described in the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*). This truststore will be used in the following cases:

Option	Description
	■ If you select the **Secure Connection** option for a CloudStreams server as described in "Defining and Managing a CloudStreams Server Target" on page 59. The truststore you specify in the **IS Truststore Name** field must contain the required SSL certificates. ■ This truststore is also used by the CloudStreams SMTP e-mail server if you select the **TLS Enabled** option while "Setting the E-mail Options for Logging Payloads and Sending Performance Monitoring Alerts" on page 31. The truststore you specify in the **IS Truststore Name** field must include a certificate in the e-mail server's certificate chain or the certificate authority of the e-mail server. ■ This truststore is also used when making a connection to a third-party OAuth 2.0-based service provider to refresh the access token (as described in "Setting the OAuth Tokens Options" on page 43).
HTTP Ports	Select one or more HTTP ports on which CloudStreams and the deployed virtual services will be available. CloudStreams is always available on the primary HTTP port. The primary HTTP port is the port specified on the Integration Server's **Security > Ports** page. You can specify additional HTTP and HTTPS ports here. To configure ports for Integration Server, see *webMethods Integration Server Administrator's Guide*. **Note:** If you specify multiple ports, the port that is reported in the WSDL is the non-primary port with the lowest number. There is only one port reported in the WSDL retrieved through CloudStreams, but all the specified ports are usable. If you are using a webMethods Enterprise Gateway (EG) Registration port to process requests from external clients, the EG port will be displayed with the EG hostname prefixed to the port, in order to distinguish it from the local internal server ports. The Enterprise Gateway Registration port must be specified for CloudStreams.
HTTPS Ports	Select one or more HTTPS ports on which CloudStreams and the deployed virtual services will be available.
Connection Factory Wire Logging	This option captures additional debug messages that show the request's HTTP content that is sent over the wire to the native provider. The messages also show the response content that was received. This can be useful when debugging connector

Option	Description
	service invocations, but there are two implications you should be aware of: ■ The `server.log` will grow quickly as it captures this verbose content, and performance will be severely impacted. Use this option sparingly. ■ Because all "cleartext" HTTP content is captured (even the content sent over SSL), be careful to avoid security issues. In addition to all request and response message content, the transport headers are also included, including Authorization headers. **Note:** Enabling wire logging will reveal sensitive data, for example, user credentials and authorization headers in the logs. It is recommended to filter out or mask the sensitive data before sharing the logs. If you disable and then re-enable the **Connection Factory Wire Logging**, in certain cases it may be necessary to disable and then re-enable any existing managed connection pools. To do this, in the Integration Server Administrator go to **Solutions > CloudStreams > Providers**, click your cloud provider's name, click the connector name, and click the **Enabled** column to disable/re-enable the connection pool. **Note:** By default, Integration Server keeps archived server log files indefinitely. Because server.log files can rapidly become large or numerous when using more verbose logging levels, you might want to limit the number of server log files that Integration Server keeps on the file system. See the *Limiting the Number of Server Log Files Kept by Integration Server* section in the *webMethods Integration Server Administrator's Guide* for more information. If you enable **Connection factory wire logging**, requests and responses containing actual data will be logged in the Integration Server logs. This log data may contain account specific or user specific data, for example, email addresses, which may be considered as personal data or personally identifiable information. If for privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), you want to delete the personal data of a user, you can manage the log files that contain the wire logging data from the file system. See the *webMethods Integration Server*

Option	Description
	Administrator's Guide for information on how to manage these logs, their related files, and specific identifiers to look for in the logs.
Handle Binary Streams	Large data configuration enables CloudStreams to send and receive large binary streams over HTTP/HTTPS. If you enable **Handle Binary Streams**, then during outbound and inbound invocations, if the stream is greater than the **Threshold Size (bytes)**, the entire stream is not stored in memory. **Threshold Size (bytes)** is applicable only if **Handle Binary Streams** is enabled. Before enabling the large data handling capability of CloudStreams, configure the TSpace properties (watt.server.tspace.*) in the webMethods Integration Server Administrator > Settings > Extended > Show and Hide Keys page. See the *webMethods Integration Server Administrator's Guide* for information on the TSpace properties. After setting these values, restart Integration Server. **Note:** Errors may appear if the watt properties are not configured properly. Increase the `watt.server.tspace.max` value in case of InsufficientSpaceException errors or increase the `watt.server.tspace.timeToLive` value incase of Connection reset by peer: socket write errors. **Keep the following points in mind regarding large data configuration:** ■ If the request/response payload is greater than the **Threshold Size (bytes)** and if the log request or response is greater than the specified **Threshold Size (bytes)**, then the request/response will neither be logged in the database nor in the Integration Server log. This is designed to prevent the Integration Server and database logs from getting flooded with large number of log messages. ■ If you want to send a large binary stream and if the back end supports HTTP chunking, ensure that you set the **Use Chunking** option in the connection configuration page to `true`. Else, heap space errors may appear even if large data handling is enabled. ■ If you want to send a large binary stream but if the back end does not support HTTP chunking, then heap space errors may appear even if large data handling is enabled. ■ Currently, **Handle Binary Streams** and **Threshold Size (bytes)** cannot be deployed using webMethods Deployer.

Setting the E-mail Options for Logging Payloads and Sending Performance Monitoring Alerts

If you use the following policy actions to send e-mail messages, you must specify the SMTP mail server to be used by CloudStreams:

- A "Log Invocation" action that you configure to send request/response payloads as e-mail attachments. See "Log Invocation" on page 171 for General Data Protection Regulation (GDPR) related information.
- A Monitoring action that you configure to send monitoring alerts as e-mail messages ("Monitor Service Performance", "Monitor Service Level Agreement" or "Throttling Traffic Optimization").

To set the CloudStreams > Administration > Email options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > Email**.

2. Click **Edit**, complete the following fields and click **Save**.

Option	Description
SMTP Host Name/IP Address	The address of the SMTP server to be used by CloudStreams for sending e-mail alerts.
Port	The port on which the SMTP server is listening.
User	The user name of the e-mail account used to log into your SMTP server.
Password	The password of the e-mail account used to log into your SMTP server.
From	The e-mail address that will appear in the From field of the event e-mail. If this field is left blank, CloudStreams generates a default e-mail address composed of the configured target name and hostname of the Integration Server, as follows: *Target-Name@hostname*
TLS Enabled	Whether to use transport-layer security (TLS). Requirements: - A keystore and a truststore must be configured in Integration Server (see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*).

Option	Description
	■ An Integration Server keystore and truststore must be specified in the **IS Keystore Name** and **IS Truststore Name** fields in **CloudStreams > Administration > General** (see "Setting the General Options" on page 26). The truststore specified in **IS Truststore Name** must include a certificate in the e-mail server's certificate chain or the certificate authority of the e-mail server.
Test Recipient	To test your connection, enter in this field an e-mail address of the person who should receive the test e-mail, and click **Test**. If the test is successful, CloudStreams sends an e-mail to the address and displays a success message. Otherwise, CloudStreams displays an error message; reconfigure your e-mail settings and try again.

Setting the Database Options for Publishing Performance Metrics and Events

CloudStreams can publish data about key performance indicator (KPI) metrics and run-time events to the CloudStreams Analytics dashboard.

CloudStreams uses the Integration Server's JDBC connection pool to publish this data. The JDBC connection pool is specified in Integration Server Administrator's **Settings > JDBC Pools** page. You must define this connection pool for use by CloudStreams as described below.

To set the CloudStreams > Administration > Database options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > Database**.

2. Click **Edit**, complete the following fields on the Database Configuration page, and click **Save**.

Option	Description
Database Publishing	Select this option to enable the database specified in the **Connection Pool Alias** to receive data. **Note:** By default, Database Publishing is disabled. If you want to use the CloudStreams Analytics feature (KPI metrics and run-time events), enable this option.
Connection Pool Alias	The Integration Server connection pool alias that is defined in the Integration Server Administrator's **Settings > JDBC Pools** page, as described in "Creating a JDBC Connection Pool for CloudStreams" on page 253.

Option	Description
	If the Integration Server connection pool alias is not yet defined, click the link to define it.
Publish Performance Metrics to database	Select this option to enable CloudStreams to publish key performance indicator (KPI) metrics to the database. For more information, see "The Key Performance Indicator (KPI) Metrics" on page 249.
Publish Interval (minutes)	The interval at which CloudStreams should publish performance metrics to the database. Default: 60 minutes. For more information, see "The Intervals for Metric Publishing" on page 33, below.
Publish Events to database (Error, Lifecycle, Policy Violation)	Select the types of events to publish to the database: ■ **Error**: An Error event occurs each time an invocation of a cloud connector service or an inbound virtual service results in an error. ■ **Lifecycle**: A Lifecycle event occurs each time the Integration Server is started/shutdown/restarted or the WmCloudStreams package is unloaded/reloaded in a running Integration Server. ■ **Policy Violation**: A Policy Violation event occurs each time an invocation of a cloud connector service violates a policy that was set for its associated virtual service.

The Intervals for Metric Publishing

CloudStreams tracks performance metrics by intervals. The interval is a period of time you set in CloudStreams, during which metrics are collected for reporting to CloudStreams.

There are two cases in which you specify an interval when measuring performance:

- When you publish performance indicator (KPI) metrics and events to the CloudStreams Analytics dashboard. In this case, you set the interval in Integration Server Administrator, in **Solutions > CloudStreams > Administration > Database**.
- When monitoring a virtual service's run-time performance with the actions "Monitor Service Performance", "Monitor Service Level Agreement", or "Throttling Traffic Optimization". In this case, you set an interval in the action's "interval" parameter.

The rules of intervals are the same in both cases.

CloudStreams only tracks metrics for the current interval. At the end of the interval, CloudStreams aggregates the metrics and reports them to CloudStreams. CloudStreams resets its counters immediately for the next interval, even if the last interval's data has not yet been reported to the database (so there is no gap between intervals when metrics are not accumulated). CloudStreams

does not calculate and aggregate metrics across intervals. If CloudStreams is shut down or the service is undeployed before the current interval expires, the performance data is discarded.

Examples of interval metric publishing

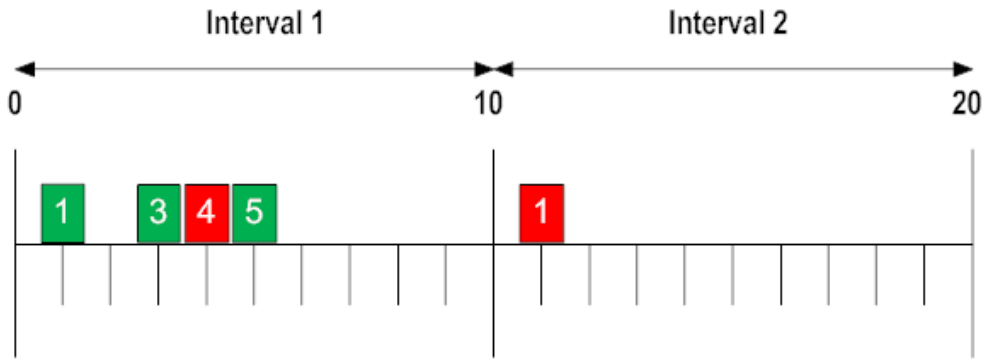
For example, suppose that the tracking interval is 10 minutes. One of the key performance indicator (KPI) metrics is “Availability”, which reports the amount of time that a service was available during the current interval, shown as a percentage. The green boxes in the illustration below indicate successful requests and the red ones indicate unsuccessful requests.

A request is considered unsuccessful when a network fault occurs or when the native service is unavailable due to the following errors:

- ConnectException
- MalformedURLException
- NoRouteToHostException
- ProtocolException
- SocketTimeoutException
- UnknownHostException
- UnknownServiceException
- Web Service not available
- The service cannot be found

In the case of a normal application-level SOAP fault, CloudStreams considers the request to be successful. In the illustration below, in Interval 1 (0 - 10 minutes) a request failed at the 4 minute mark, followed by a successful request at the 5 minute mark. Therefore, CloudStreams considers the interval between 4 and 5 minutes to be service downtime (even though this may not be accurate). So in this case, for Interval 1 the availability is 9/10 (90%). In the case of Interval 2, only one request was sent, and it failed at the 1 minute mark. Therefore, CloudStreams considers the time between 1 minute to the end of the interval as service downtime. So the time between the start of Interval 2 (the 10 minute mark) to the failed request is service uptime (1 minute); the availability is 1/10 (10%) for Interval 2. At the end of the interval, CloudStreams resets the KPI metrics.

The “Availability” status of a service is transferred to the beginning of the next interval. When CloudStreams starts up, it is assumed that the services are available. If a service is unavailable at the end of interval 1, and no activity occurs for several intervals, the service's availability is 0 during the intervals when no activity occurred.



The Virtual Services Option

This option displays a list of:

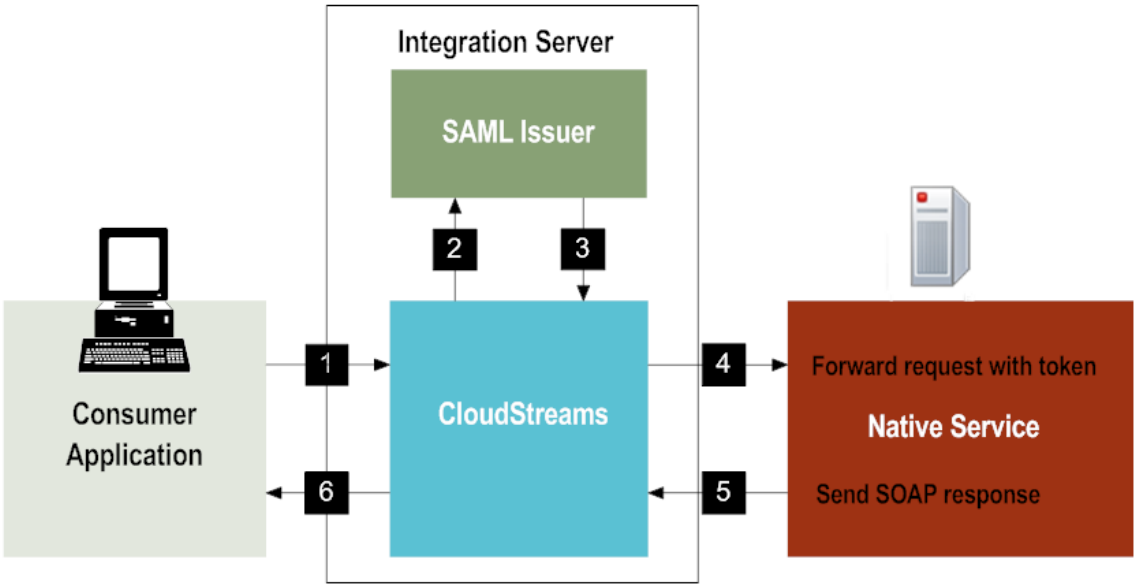
- The default connector virtual services (**WmCloudStreams.SoapVS** and **WmCloudStreams.RestVS**) as well as any other user-defined connector virtual services in the WmCloudStreams package.
- All virtual services defined in the WmCloudStreams package that are currently deployed to a CloudStreams server target.

To view the list, select **Solutions > CloudStreams > Administration > Virtual Services** in Integration Server Administrator.

Setting the STS Options

CloudStreams can act as a Security Token Service (STS) client. The following illustration shows what happens at run time.

CloudStreams as an STS Client



Step	Description
1	The user's client sends a SOAP request with SAML authentication information to CloudStreams. Integration Server authenticates the incoming request.
2	■ CloudStreams sends a WS-Trust RST to the STS to request a SAML v2 token. ■ CloudStreams sends the `<OnBehalfOf>` element that contains the authenticated user name to the STS.
3	The SAML Issuer sends the SAML v1/v2 assertion to CloudStreams.
4	CloudStreams forwards the SOAP request (along with the SAML assertion) to the native service. CloudStreams also uses the IS keystore and signing alias you specified to sign the SAML token and the request body before sending the request to the native service. Also, if you have configured the predefined Java service pub.cloudstreams.security.ws.AddSamlSenderVouchesTokento add a timestamp in the outbound request, CloudStreams will sign the timestamp as well.
5	The native service sends a SOAP response to CloudStreams.
6	CloudStreams sends the response to the user's client.

Use the following procedure to enable CloudStreams to act as a Security Token Service (STS) client.

To set the CloudStreams > Administration > STS options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > STS**.

2. You can use Integration Server's default STS, **DefaultSTS**. To view its default parameters, click its name.

3. Alternatively, you can to use a third-party STS that has been defined in the Integration Server (as described in the *Web Services Developer's Guide*, in the section *Securing Web Services Using Policies Based on WS-SecurityPolicy*). To use a third-party STS, click **Add new STS configuration** and complete the Add STS Configuration page as follows.

Option	Description
Name	A unique name for the STS being configured. If this value is changed after creating an STS, the previous STS configuration will be deleted and replaced with the new one.
Endpoint	The STS endpoint to which the WS-Trust request will be sent by CloudStreams to obtain the SAML token.
Token Type	The type of token that CloudStreams should request from the STS. Value can be SAML_11 or SAML_20. **Note:** Currently the default Integration Server STS only supports issuing SAML 2.0 Sender-vouches tokens.
WS-Trust Version	The version of WS-Trust that CloudStreams should use to send the RST to the SAML Issuer. Value can be VERSION_05_02 or VERSION_05_12.
Time-To-Live (TTL)	Indicates the time-to-live value in seconds that will be specified in the RST. If not specified, the default is 300 seconds (5 minutes).
Keystore	Select a keystore that has been configured in Integration Server. To configure a keystore in Integration Server, see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*.
Sign Request/Signing Alias	Select **Sign Request?** if the STS requires a signed request, and specify the signing alias in the **Signing Alias** field.
Encrypt Request/Encryption Alias	Select **Encrypt Request?** if the STS requires an encrypted request, and specify the encryption alias in the **Encryption Alias** field.

Option	Description
HTTP Basic Authentication-Username/Password	If the STS requires authentication, enter the HTTP Basic Authentication username and password.
WS-Security Username Token-Username, Password, and Password Type	The WS-Security username token to send to the STS. Also specify the username's password and password type (None, Text or Digest).

4. If you selected Integration Server's default STS (**DefaultSTS**), edit the default STS's configuration file to specify the keystore and alias so that the STS can sign the SAML assertion that it is issuing.

   The configuration file is:

   IntegrationServer_directory\instances*instance_name*\config\security\saml\esb_sts.xml

   The contents of the file are shown below. Use the comments as a guide to configure this file for your system.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- This configuration file is used to configure the IntegrationServer token
issuer that generates the SAML Sender Vouches token for CloudStreams outbound
requests -->

<IDataXMLCoder version="1.0">
<record javaclass="com.wm.data.ISMemDataImpl">

<!-- IssuerName - will be used as the IssuerName for each SAML token issued by
this Service; the default value is ESB_STS -->
<value name="IssuerName">ESB_STS</value>

<!-- IssuerKeystoreAlias - specify an Integration Server Keystore Alias that
contains the private keys that can be used to sign the generated SAML
Assertion -->
<value name="IssuerKeystoreAlias">STS</value>

  <!-- IssuerKeyAlias - the name of the key alias within the
IssuerKeystoreAlias that points to the private key files -->
<value name="IssuerKeyAlias">sts</value>

  <!-- TimeToLiveSeconds - how long in seconds the generated token should be
valid? the default is 300 seconds (i.e. 5 minutes) from the time of token
creation -->
<number name="TimeToLiveSeconds" type="java.lang.Integer">300</number>

  </record>
</IDataXMLCoder>
```

5. Configure the desired virtual services so they can use the STS. To do this, write an IS service that includes the predefined Java service pub.cloudstreams.security.ws.AddSamlSenderVouchesTokenand invoke this IS service in the In Sequence step of the virtual service. This service must reference an STS that is identified in

CloudStreams. For details about the AddSamlSenderVouchesTokenservice, see "Using the Security API in IS Services" on page 88.

Setting the Service Fault Configuration Options

Note:
The service fault configuration options are intended only for virtual services, and not for Connector Virtual Services. For information about error handling for Connector Virtual Services, see "The Error Sequence Step (Connector Virtual Service, SOAP)" on page 120 or "The Error Sequence Step (Connector Virtual Service, REST)" on page 152.

You can use these options to configure global error responses for all virtual services.

Alternatively, you can configure error responses for virtual services individually, in the services' "Error Sequence" step, as described in "The Error Sequence Step (Connector Virtual Service, SOAP)" on page 120 or "The Error Sequence Step (Connector Virtual Service, REST)" on page 152.

The precedence of the error messaging instructions is as follows:

- If you create an "Error Sequence" step for a virtual service, the error messaging step takes precedence over any settings on the Service Fault Configuration page.
- If you do not create an "Error Sequence" step for a virtual service, the settings on the Service Fault Configuration page take precedence.

To set the CloudStreams > Administration > Service Fault Configuration options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > Service Fault Configuration**.

2. Click **Edit**, complete the following fields and click **Save**.

Option	Description
Default Fault Response	When you select this option, CloudStreams returns the following fault response to the consuming application: `CloudStreams encountered an error:$ERROR_MESSAGE while executing operation:$OPERATION service:$SERVICE at time:$TIME on date:$DATE. The client ip was:$CLIENT_IP. The current user:$USER. The consumer application:$CONSUMER_APPLICATION".` **Note:** When $CLIENT_IP is used, CloudStreams will replace $CLIENT_IP with the IP address of the client. For privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), click **Edit** and then delete the *"The client ip was:$CLIENT_IP."* string from the Service Fault Configuration template.

Option	Description
	This fault response is returned in both of the following cases: ■ When a fault is returned by the native service provider. In this case, the `$ERROR_MESSAGE` variable in the fault response will contain the message produced by the provider's exception that caused the error. This is equivalent to the `getMessage` call on the Java Exception. This maps to the faultString element for SOAP 1.1 or the Reason element for SOAP 1.2 catch. For REST service calls, the message is returned inside an `</Exception>` tag. CloudStreams discards the native service provider's fault and does not return this content to the web service caller since it could be considered a security issue, especially if the native provider is returning a stack trace with its response. ■ When a fault is returned by internal CloudStreams exceptions (policy violation errors, cloud connection errors and cloud connector service errors). In this case, the `$ERROR_MESSAGE` variable will contain the error message generated by CloudStreams. The default fault response contains predefined fault handler variables (`$ERROR_MESSAGE`, `$OPERATION`, etc.), which are described in "The Fault Handler Variables" on page 87. You can customize the default fault response using the following substitution variables, where CloudStreams replaces the variable reference with the real content at run time: ■ The predefined context variables listed in "The Predefined Context Variables" on page 97. ■ Custom context variables that you declare using the CloudStreams API (see "The API for Context Variables" on page 100). **Note:** If you want to reference a custom context variable that you have already defined in a context-based routing rule (as opposed to one you have declared using CloudStreams API for context variables), then you must add the prefix `$mx` to the variable name in order to reference the variable. For example, if you defined the variable `TAXID`, you would reference it as `$mx:TAXID`.
Send Native Provider Fault (when available)	When you select this option, CloudStreams sends the native service provider's fault content, if available. CloudStreams will send whatever content it received from the native service provider.

Option	Description
	If you select this option, the **Default Fault Response** is ignored when a fault is returned by the native service provider. (Faults returned by internal CloudStreams exceptions will still be handled by the **Default Fault Response** option.)

Note:
Unlike with SOAP specifications, there is no agreed upon format to suggest an error condition for REST services (that is, there is no element nested in a `</soap:Fault>` element nested in a `</soap:Body>`). CloudStreams assumes that REST services will follow HTTP conventions and return responses with return codes in the 200-299 range when service calls are successful. This is the only way CloudStreams can determine that a native provider's response should be interpreted as a failure.

Setting the Consumers Options

Use this option to define the consumer applications that should be allowed to access CloudStreams.

On this page, specify the precise values for the consumer identifier(s) that you specified for the Identify Consumer action in the policies of your virtual services.

To set the CloudStreams > Administration > Consumers options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > Consumers**.

2. Click **Add New Consumer**, complete the following fields and click **Save**.

Option	Description
Name	Assign a name for the consumer application.
Description	Enter a description for the consumer application.
IP Addresses	Use this field when the Identify Consumer action is configured to identify consumer applications by IP address. Enter one or more IP addresses (or ranges of addresses) to identify consumers. IPv4 and IPv6 addresses are supported. If entering an IPv4 address, enter a 4-byte IP address. Note that: ■ To specify an individual IP address, type the address in the **From** field. The Identify Consumer action will identify only those requests that originate from the specified IP address. ■ To specify a range of IP addresses, type the lowest IP address in the **From** field and the highest IP address in the **To** field. For

Option	Description
	example, the values 192.168.0.0 and 192.168.0.10 indicate that requests originating from any IP address that lies between the specified range will be identified by CloudStreams. If entering an IPv6 address, use the 128-bit IPv6 format. For example: `1234:5678:9ABC:DEF0:1234:5678:9ABC:DEF0`. To specify additional IP addresses, use the plus button to add more rows.
Identification Token	Use this field when the Identify Consumer action is configured to identify consumer applications by an identification token. Specify one or more of the following tokens. To specify additional tokens, use the plus button to add more rows. ■ **Host Name**: To identify consumers based on a specified host name, type the host name (for example, `pcmachine.ab.com`) in the **Name** field. CloudStreams will identify only those requests that originate from the specified host name. ■ **HTTP Authorization Token** : To authorize consumers based on the user name transmitted in an HTTP authorization user token, type the user name (for example, `testuser123`) in the **Name** field. The application asset will identify only the requests that contain the specified user name encoded and passed in the HTTP user token in the HTTP Authorization header. Authentication is handled by LDAP or another external authentication mechanism. ■ **WS-Security Authentication Token**: To authenticate consumers based on the user name transmitted in the SOAP or XML message header (HTTP body), type the user name (for example, `userwss`) in the **Name** field. CloudStreams will identify only the requests that contain the specified user name passed in the SOAP or XML message header. Authentication is handled by LDAP or another external authentication mechanism. ■ **A custom identification token (XPath)**: To identify consumers based on the result of applying an XPath expression on the SOAP or XML message or request, enter the XPath expression in the **Name** field. For example, typing `//*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='echoInt']/*[local-name()='echoIntInput'][.='2']` in the Name field will identify the requests that contain the XPath and the consumers. ■ **User ID**: For connector virtual services only. Identifies consumers based on a list of users who are registered in the Integration Server on which CloudStreams is running.

Option	Description
X.509 Certificate	Use this field when the Identify Consumer action is configured to identify consumer applications by an X.509 v3 consumer certificate. Specify the path of a certificate file on the server.

Setting the OAuth Tokens Options

If you will be setting the **Authentication Type** field to **OAuth** on the Streaming Providers page (as described in "Setting the Streaming Providers Options" on page 46), you need to set the options here, on the OAuth Access Tokens page.

If OAuth will be used as an authentication mechanism, you must create an alias containing OAuth details. For example, when a Streaming Provider is created that will use OAuth, the provider details include an OAuth alias. When OAuth is chosen on the Streaming Providers page, an alias must be selected from a drop-down list of OAuth Token aliases that are currently defined. At run time, the OAuth alias is used to lookup parameters, such as Access Token, that are used to establish the connection with the provider.

OAuth is a scheme to limit the need for user credentials when operating on behalf of a resource owner. Instead of using the owner's credentials at run time, an Access Token is used. Prior to run time, the owner and user engage in a "dance" with a Resource Server and an Authorization Server to obtain an Access Token that will grant the user limited access to resources controlled by the owner. The precise steps in the dance vary with OAuth version, provider (Salesforce versus Twitter) and OAuth grant scheme ("two-legged dance" versus "three-legged dance"). Given this variation and the complexity of this process, CloudStreams assumes that the proper OAuth fields are obtained, and stores the tokens and other properties for use at run time.

Some of the information that is stored is considered to be sensitive and is stored securely, just as a password would be.

Currently, two OAuth versions are supported, v1.0a and v2.0. The parameters that are stored with an alias are different in the two versions, though there is also significant overlap.

To set the CloudStreams > Administration > OAuth Tokens options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Administration > OAuth Tokens**.

2. Click **Add New Token**, complete the following fields, and click **Save**.

Option	Description
Name	A unique string to identify a collection of alias token parameters.
Provider	Identifies the OAuth provider (such as Salesforce or Twitter).
OAuth Version	Select the OAuth version, v1.0a or v2.0.

Option	Description
Consumer ID	The 'Consumer Key' issued by the Service Provider and used by the consumer to identify itself to the Service Provider. In OAuth 2.0, this is a 'client identifier' issued to the client to identify itself to the authorization server. Also referred to as the “Client ID”.
Consumer Secret	In OAuth 2.0, this is an optional field. This is a secret used by the Consumer to establish ownership of the "Consumer Key" and is a secret matching to the "client identifier". It is also referred to as the "Client Secret".
Access Token	The token used for authentication, instead of other credentials. This token is issued by the Authorization Server. The process for obtaining a token varies with version, vendor and OAuth grant type. For OAuth 2.0, CloudStreams will use the keystore and truststore specified in the **CloudStreams > Administration > General** tab when making a connection to the third-party OAuth 2.0-based service provider to refresh the access token. The refresh URL is almost always SSL-based, so the truststore specified in CloudStreams will be used during the SSL handshake to verify the provider certificate. If the provider requires two-way SSL authentication, then the keystore specified in CloudStreams will be used to provide the client certificates. For details about the keystore and truststore specified in CloudStreams, see “Setting the General Options” on page 26. If a truststore is *not* specified in the **CloudStreams > Administration > General** tab, CloudStreams will use the default truststore for the jvm (cacerts). This might be a problem if you have defined the default truststore to only contain specific certificates, because if that alias is defined, the well-known certificates will not be included when defining the trust. In this case, make certain the truststore alias keystore includes the certs necessary to trust the URLs defined in this field too.
Access Token Secret	Issued with the v1.0a Access Token only. A secret used by the Consumer to establish ownership of a given 'Access Token'. Secrets are used to sign the requests.
Instance URL	This is an optional field and is used to specify a runtime host, if applicable. This may be required in some back ends like Salesforce.
Refresh Access Token	Option to refresh the 'Access Token'. OAuth 2.0 access tokens typically have a very short lifetime. When an access token expires, the OAuth profile does not automatically refresh the expired access token. Select this option if you want an expired access token to be

Option	Description
	refreshed automatically. If you select this option, you must also specify **Refresh Token** and **Refresh URL**. The access token is refreshed whenever the session expires. Session expiration is handled according to the setting of the Session Management property in your connection. Note that if Session Management is set to “none”, then you must manually modify the access token in the OAuth alias. (The **Refresh Access Token** option will not be applicable in this case.) If you want to refresh the 'Access Token' automatically, set Session Management to either 'fixed' or 'idle'. The Timeout value should be based on the backend settings.
Refresh Token	Issued with the OAuth v2.0 access token only. Required only if **Refresh Access Token** is enabled. A token used by the client to obtain a new access token without having to involve the resource owner.
Refresh URL	Available only if **Refresh Access Token** is selected. The provider specific URL to refresh an 'Access Token'. This is required when 'Refresh Access Token' is enabled.

Refresh URL Request	Options for sending the parameters, for example, refresh_token, grant_type, client_id, and client_secret required by an access token refresh (HTTP POST) request.
	
	The available options are:
	
	■ **URL Query String** - The refresh request parameters, for example, refresh_token, grant_type, and so on, and their values are sent as query strings in the URL of the POST request.
	
	Example:
	
	`www.examplebackend.com/o/oauth2/token?grant_type=refresh_token&client_id=842428530070-pubfebfgfqkgj6t54m4ns6&client_secret=4adQT95cAtUxWINbDxGP9SJ4&refresh_token=1%2Fn072P4BXpuNObjCLUtiZTc4fMH6YersmxBIv8QN3bhw`
	
	■ **Body Query String** - The refresh request parameters, for example, refresh_token, grant_type, and so on, and their values are sent as query strings in the body of the POST request.
	
	Example:
	
	`POST /o/oauth2/token HTTP/1.1`
	`Host: accounts.backend.com`
	`Content-length: 163`

Option	Description

	`content-type: application/x-www-form-urlencoded`
	
	`client_secret4adQT95cAtUxWINbDxGP9SJ4&grant_type=refresh_token&refresh_token=1%2Fn072P4BXpuNObjCLUtiZTc4fMH6YersmxBIv8QN3bhw&client_id=407408718192`
	
	■ **Custom ESB Service** - Use this option if the back end requires some refresh requests in a custom format, for example, requests which need more parameters than the ones specified by OAuth v2.0, or the back end uses some custom way of organizing parameters, or expects some other HTTP method request (other than POST). If you select this option, you must specify the name of your custom service in the 'Service Name' field.

Service Name	User implemented service for refreshing the 'Access Token'. This is required when the **Refresh URL Request** is specified as **Custom ESB Service**. This service should strictly conform to the following specification: `-wm.cloudstreams.service.common.lookup.specs: oauthTokenRefreshServiceSpec`

The Providers Options

Use these options to:

- Enable/disable/delete/configure CloudStreams cloud connectors.
- Create/enable/disable/delete/view/copy managed cloud connections.

For details, see the documentation specific to your CloudStreams cloud connector (for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*).

The Streaming Options

In Integration Server Administrator, you can set the following options under **Solutions > CloudStreams > Streaming**:

- "Setting the Streaming Providers Options" on page 46
- "Setting the Streaming Subscribers Options" on page 49

Setting the Streaming Providers Options

To set the CloudStreams > Streaming > Providers options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Streaming > Providers**.

2. Click **Add New Provider**, complete the following fields and click **Save**.

Option	Description
Provider Name	Specify a unique name for this streaming service provider. This is a required field and is used by the Subscriber configuration to refer to it.
Client Type	The type of client that will be used by CloudStreams to provide the streaming functionality. There are two types of clients: ■ **Comet**: This is a client implementation that uses the Bayeux HTTP protocol (using the CometD library) to communicate with the streaming service. This client should be used if the streaming provider explicitly indicates that they support the Bayeux protocol for their streaming functionality. For example, Salesforce uses this mechanism for their streaming functionality and hence Salesforce integrations must use this client type. **Note:** Underlying protocols that provide streaming capability are public protocols which run over HTTP like Bayeux. These protocols are not guaranteed protocols. They do not ensure delivery of messages in a guaranteed way. If there are network, proxy, or firewall connectivity related issues or delays, you may not receive the messages. In such cases, investigate the network and connectivity. Before using streaming capabilities, it is important to understand this underlying limitation of the protocol and the SaaS backend. ■ **HTTP**: An HTTP client type is used for HTTP-based streaming service providers that use a long-lived HTTP connection to send data periodically to the client connected to them. For example, Twitter uses this mechanism to send updates made to their streaming API endpoints (Public, User and Site streams).
API Version	Optional. The version of the streaming API supported by the provider service.
Streaming API Endpoint	Points to the endpoint of the streaming provider. For example, when configuring a Salesforce streaming endpoint, you must specify the URL that corresponds to the streaming API, such as: https://na9-api.salesforce.com/cometd/25.0

Option	Description
	Do not specify the actual subscription channel/topic name here as part of the URL because that is subscriber configuration-specific.
Connection Timeout (ms)	Specify the streaming connection timeout value, in milliseconds. If unspecified, this value defaults to a system default.

Note:
It is observed that due to network, proxy, or firewall connectivity related issues, the default timeout values may not be enough in your environment. In that case you should check what are the right timeout settings based on your network conditions and set those values here. Due to latency and other issues, some networks may take a long time to exchange messages. In such cases you may see timeout errors. To avoid that, increase the timeout values and see what values fit your organization. This is applicable for all outbound and inbound timeout configurations.

Read Timeout (ms)	Specify the read timeout value after a connection has been established with the provider. This indicates the maximum time before which a data packet must be read from the endpoint before a timeout occurs.
Truststore Alias	Specify the IS truststore alias to use if the endpoint is SSL-based and the provider certificate must be validated as part of the SSL handshake. By default, if no alias is configured, then all server certificates are trusted during SSL handshake. The alias must be configured in Integration Server Administrator. To configure Integration Server truststores, see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*.
Keystore Alias	Specify the IS keystore alias to use if the endpoint is SSL-based and the CloudStreams certificate must be provided to the server endpoint as part of the SSL handshake. By default, if no alias is configured, then no client certificate is sent during SSL handshake. The alias must be configured in Integration Server Administrator. To configure Integration Server keystores, see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*.
Validate Certificate	If set to true, the client will validate the provider's certificate for the SSL handshake. Default: False.
Authentication Type	Specifies the authentication information that must be sent to the streaming provider when a subscription to a topic or channel is defined. Choose one of the following ways to provide the authentication information:

Option	Description
	■ **Basic**: Basic refers to HTTP Basic Authentication. This option can be used if the provider requires or supports HTTP Basic authentication using a username and password. ■ **OAuth 1.0a** and **OAuth 2.0**: Select an OAuth alias that was defined on the OAuth Access Tokens page (see "Setting the OAuth Tokens Options" on page 43). At run time, CloudStreams will use the OAuth configuration information to retrieve the access token to be sent to the provider in the HTTP request. ■ **ESB Callback**: This option can be used when the above two means of authentication are not sufficient. For example, if the provider does not support Basic authentication or OAuth tokens, but instead requires another form of authentication information as part of the `Authorization` header, then you can provide another form of authentication in a user-defined ESB service. Specify the service name in this field in the following form: *`folder1.folder2:serviceName`*. The only requirement for this ESB service is that it should specify the value for the `Authorization` header in the `authorization.header` pipeline variable. CloudStreams will use the value present in this pipeline variable to send it as part of the `Authorization` header value. You should *not* include the header name (`Authorization`) in the value set in this variable.

Setting the Streaming Subscribers Options

CloudStreams supports the following kinds of streaming client implementations:

- Client implementations based on the Comet library, which works with streaming services using the Bayeux protocol (for example, Salesforce).
- An HTTP client type, which is used for HTTP-based streaming service providers that use a long-lived HTTP connection to send data periodically to the client connected to them. For example, Twitter uses this mechanism to send updates made to their streaming API endpoints (Public, User and Site streams).

To set the CloudStreams > Streaming > Subscribers options

1. In Integration Server Administrator, select **Solutions > CloudStreams > Streaming > Subscribers**.
2. Click **Add New Subscriber**, complete the following fields and click **Save**.

Option	Description
Enable Subscriber	Enables or disables the subscriber configuration. If the subscriber configuration is disabled, then no streaming notifications will be received by CloudStreams until it is enabled again.
Subscriber Name	A name that uniquely identifies a subscriber.
Provider	One of the streaming providers previously configured in "Setting the Streaming Providers Options" on page 46. The provider names are listed in a drop-down field, and each subscriber must choose the provider configuration to use in order for the streaming subscription to work.
Channel Endpoint	The topic/channel name or the endpoint address for the subscription request, depending on the streaming provider's client type, as follows: ■ If the streaming provider's client type is **Comet**, specify the topic or channel name for the subscription request. This is usually of the form `/channel_name`. For example, for Salesforce it can something like `/topic/InvoiceUpdate`. You must not specify the complete subscription endpoint address (of the form `http://server:port/...`), because that information is part of the provider configuration information. The channel endpoint must refer to the actual channel name only. This endpoint will be appended to the provider endpoint before the actual streaming connection is made to the provider. ■ If the streaming provider's client type is **HTTP**, specify the endpoint address without the actual stream/channel name included. For example, Twitter's Public Stream address is: `https://stream.twitter.com/1.1/channelEndpoint`
Content Type	The HTTP Content-Type header value. Not required if the Streaming Providers **Client Type** field is set to **Comet**.
HTTP Method	The HTTP method to use for the streaming request made to the service provider. Not required if the Streaming Providers **Client Type** field is set to **Comet**.
Request Body	The optional request entity or message contents to send to the service endpoint. This is only applicable for the POST, PUT, OPTIONS and PATCH HTTP methods; the other HTTP methods do not require a message body. Not required if the Streaming Providers **Client Type** field is set to **Comet**.
HTTP Headers	Additional HTTP headers (name-value pairs) to be sent to the service endpoint (optional). Not required if the Streaming Providers **Client Type** field is set to **Comet**.

Option	Description
Destination Type	The type of destination to which the streaming subscription response will be delivered when a notification is received from the provider: ■ **ESB Service**: An Integration Server service that will be invoked when a streaming notification is received. The **ESB Service Destination Configuration** section appears if you select **ESB Service** as the **Destination Type**. **Service Name**: The fully qualified service name comprises of two parts: a folder identifier and the service name. The folder identifier consists of one or more folder names. The service name is a single name of the service and must be of the fully qualified form *folder1.folder2:serviceName*. For more information, see "Creating an ESB Service as a Streaming Subscriber Destination" on page 51. **Run As User**: The user name you want Integration Server to use when running the service. Click the search icon to search for and select the user. A user can be selected from the local or central directory. Integration Server runs the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an Access Control List (ACL), ensure that you specify a user that is allowed to invoke the service. ■ **Journal Log**: The IS journal log to which the streaming subscription response contents will be logged when a notification is received from the provider. You can specify the logging level in the **Log Level** field.

Creating an ESB Service as a Streaming Subscriber Destination

When you select the **ESB Service** option for the **Destination Type** field for a streaming subscriber, you specify an IS service that will be invoked when a streaming notification is received. Following is guidance on how to write such an IS service.

ESB Services for the Client Type of Comet

The response is present under an IData variable in the pipeline, with a key whose value is the same as the name of the subscriber. For example, if the subscriber is named `MySalesforceTopic`, then there will be a key in the pipeline with the same name.

The IData structure is:

```
[subscriber_name]
   message
```

```
        channel.name (if present)
        channel.endpoint
        data.json
        comet.msg
```

This IData variable contains the actual subscription response contents within the `message` pipeline variable, which contains the following pipeline variables:

- `channel.name`: The name of the channel over which the subscription response arrived.
- `channel.endpoint`: The endpoint value of the channel.
- `data.json`: The streaming response contents in JSON format (note that this is only applicable if the provider client type is Comet).
- `comet.msg`: The actual CometD message instance value. The type of the value is: `org.cometd.bayeux.Message`.

ESB Services for the Client Type of HTTP

The response is present under an IData variable in the pipeline, with a key whose value is the same as the name of the subscriber. For example, if the subscriber is named `MyTwitter`, then there will be a key in the pipeline with the same name.

The IData structure is:

```
[subscriber_name]
   message
        response.data
        channel.endpoint
```

This IData variable contains the actual subscription response contents within the `message` pipeline variable, which contains the following pipeline variables:

- `response.data`: A String value representing the HTTP response contents.
- `channel.endpoint`: The value corresponds to the subscriber destination from which the response was received.

For HTTP streaming providers, if an ESB service is configured to consume incoming notification messages, the service may get triggered frequently as the back end may be sending periodic control messages to ensure connectivity. In such cases, the ESB service which consumes the notifications should be designed to ignore such control messages. The contents of the periodic connectivity messages and their frequency may vary from back end to back end. For more information, refer the back end specific streaming documentation, which provides information on how streaming notifications and control messages function for a given back end.

Streaming subscriber behavior in case of an error

If a streaming subscriber runs into an error, it logs an error message and using the underlying CometD client, the streaming subscriber automatically retries the connection. If connectivity is successfully established, a fresh subscription request is sent whenever a meta/handshake event occurs. Streaming notification messages are received as before.

This behavior is observed in the case where the streaming provider **Authentication Type** is selected as **ESB Callback**.

Connection Options

CloudStreams supports the latest versions (1.1 and 1.2) of the Transport Layer Security (TLS) standard.

For increased security, SaaS providers have started using the latest TLS versions, for example, TLS Version 1.2. Some of these providers may not take care of CloudStreams requests with an older version, for example, TLS v1.0.

To use TLS Version 1.1 and 1.2 for CloudStreams connection requests, from the **Settings** menu in Integration Server Administrator, click **Extended** and add `watt.net.ssl.client.useJSSE=true` in the **Extended Settings** pane to use the latest TLS version. It is recommended to restart the Integration Server. In most cases, only this property is required for SaaS providers like Salesforce. In some cases, where you want to force usage of any other protocol, for example, SSLv3, or a particular version of TLS properties, use the following properties:

- watt.net.jsse.client.enabledProtocols
- watt.net.ssl.client.handshake.minVersion

See the *webMethods Integration Server Administrator's Guide* for desired values of these properties.

Configuring a connection trust store

For some SaaS back ends, the underlying JVM trust store may have the required certificates and you may not need to follow this procedure. But for back ends that have certificates that are not part of the JVM trust store, create and apply a new trust store in Integration Server with the certificates of the SaaS back end. This is a standard practice for setting up secure exchange of certificates. If the trust store is not present, in such cases, certificate related errors will appear.

To set up secure exchange of certificates, create a JKS trust store with the certificates of the SaaS back end. The trust store should contain all the certificates present in the certificate chain. Apply the trust store at **Security > Keystore** on the Integration Server Administrator screen, and use that trust store alias in your CloudStreams connection in the **Trust Store** field.

To know more about creating trust stores and applying them in Integration Server, see the *Securing Communications with the Server* section in the *webMethods Integration Server Administrator's Guide*. Also see the *Trust store Alias* section in the relevant CloudStreams Provider document to find the connection's advanced property, for applying the trust store in the connection.

Note:
You can also create trust stores using publicly available tools.

If connection enablement or service execution fails with time out related errors, for example, "Read Timeout" or "Timeout waiting for a connection", configure the CloudStreams connections for time out values. In case the network is slow or the back end processing takes longer than usual, increase the **Connection Timeout** and the **Socket Read Timeout** values to three or four minutes. Based on the server responsiveness and network conditions, you may have to further increase or decrease this value. If you specify 0, the connection waits indefinitely but a value of 0 should be used only

for debugging purposes and not in production environments because it can indefinitely block a connection. The correct time out values are not fixed and may vary based on the SaaS provider, load, network responsiveness, latency, and various other factors. See the *webMethods CloudStreams FAQ and Troubleshooting Guide* for information on the errors.

3 Virtual Services

Overview

Following is a summary of the four high-level steps for creating virtual services:

1. "Define a CloudStreams Server Target" on page 56.
2. "Create a CloudStreams Governance Project" on page 56.
3. "Create the Virtual Services" on page 56.
4. "Create Policies for the Virtual Services" on page 58.

Define a CloudStreams Server Target

A CloudStreams *server target* specifies a CloudStreams server to which you will deploy virtual services and connector virtual services. You define CloudStreams server targets using Software AG Designer. You can create one or more server targets.

Create a CloudStreams Governance Project

You need to create a CloudStreams Governance project in which you will create the virtual services and their policies, plus any custom connector virtual services and their policies. You create projects in your local file system, using the CloudStreams Development plug-in provided by Designer. You can create one or more projects.

Create the Virtual Services

A *virtual service* runs on a CloudStreams server and acts as the consumer-facing proxy for a native service that runs in a SaaS application or in an on-premise application. Service requests sent from a service consumer go to a virtual service hosted on the CloudStreams server for processing rather than directly to the service provider. You should create a virtual service for each SaaS service you want to expose to consumers.

There are two types of virtual services: *virtual services* and *connector virtual services*.

- In the inbound processing scenario, when a SaaS application sends a service request, a virtual service performs security checks and other user-defined processing before sending the request to the on-premise application.
- In the outbound processing scenario, when an on-premise application sends a service request, a special kind of virtual service is used, called a connector virtual service. A *connector virtual service* handles the SaaS provider's responses and logs the request/responses and their payloads.

  CloudStreams provides a default connector virtual service for each metadata handler (one for the SOAP handler and one for the REST handler). You cannot modify these default services, which are located in the WmCloudStreams package. Alternatively, you can create additional connector virtual services with custom policies.

You create both types of virtual services using the CloudStreams Development plug-in provided in Designer, and then you deploy them to a CloudStreams server.

Both types of virtual services can have *policies*, which provide governance capabilities for the services; policies are discussed later in this overview.

The Processing Steps of a Virtual Service

When you create a virtual service, you can configure the following processing steps for the service:

- The **In Sequence step**, which you configure to manipulate the request messages. This step can include the following sub-steps:
  - The **Entry Step** (required), which specifies the protocol (HTTP or HTTPS) of the requests that the virtual service will accept, and for REST services it also specifies the HTTP methods that the virtual service should be allowed to perform on a REST resource.
  - The **Transform step** (optional), which performs an XSLT message transformation on the request message before the virtual service submits it to the native service.
  - The **Invoke IS Service step** (optional), which pre-processes the request message before the virtual service submits it to the native service.
  - The **Routing Rule step** (required), which specifies how the virtual service will route the requests to the native service endpoint. There are four ways to route HTTP or HTTPS requests:
    - "Straight Through" routing (to route requests directly to the native service endpoint).
    - "Context-Based" routing (to route specific types of messages to specific endpoints according to context-based routing rules).
    - "Content-Based" routing (to route specific types of messages to specific endpoints based on specific values that appear in the request message).
    - "Load Balancing" routing (to distribute requests among multiple endpoints).
- The service's **Out Sequence step** (optional), which you configure to manipulate the response messages. This step can include the following sub-steps:
  - The **Transform step** (optional), which specifies how the response message from the native service provider is to be transformed before the virtual service returns it to the consuming application.
  - The **Invoke IS Service step** step (optional), which pre-processes the response message before the virtual service returns it to the consuming application.
- The service's **Error Sequence step** (required). CloudStreams returns a default fault response to the consuming application, which you can customize with context variables. This fault response is used for faults returned by the native service provider as well as faults returned by internal CloudStreams exceptions (policy violation errors, cloud connection errors and cloud connector service errors). In addition, you can:

- Choose whether or not to send the native service provider's service fault content, or just send the response message.
- Invoke IS services to pre-process or post-process the error messages.

The Processing Steps of a Connector Virtual Service

A connector virtual service can contain similar processing steps as a virtual service.

Create Policies for the Virtual Services

You should create policies that apply to one or more virtual services. A *policy* is a sequence of actions that is carried out by CloudStreams when a consumer requests a particular service through CloudStreams. You create policies using the same editor you use to create virtual services.

A policy for a virtual service can include the following kinds of actions:

- **WS-SecurityPolicy 1.2 actions**, which include authentication actions and XML security actions for the inbound requests.
- **Monitoring actions**, to monitor run-time performance conditions for virtual services, and to optimize server traffic (and to send alerts when the conditions are violated).
- **Additional actions**, to identify/authenticate consumers, to validate request/response messages against an XML schema, and to log request/response payloads.

For complete details, see the chapter "Policies" on page 159.

Policies for Connector Virtual Services

Each default connector virtual service has a default policy, which logs all request/response payloads to a database. You cannot modify the default connector virtual service or its policy. Alternatively, you can create additional connector virtual services with custom policies.

If you create a connector virtual service with a custom policy, you can only include the actions in the "Monitoring" or "Additional" action categories; you cannot include the "WS-SecurityPolicy 1.2" actions (except the "Require SSL" action). For example, you might want to create a custom policy that monitors run-time performance, customizes how the service invocations are logged, validates response messages against an XML schema, or optimizes server traffic.

The procedures to perform the above tasks (except for creating policies) appear below, in the following sections:

- "Defining and Managing a CloudStreams Server Target" on page 59.
- "Creating a CloudStreams Governance Project" on page 60.
- "Creating a New Virtual Service (SOAP)" on page 62.
- "Creating a New Connector Virtual Service (SOAP)" on page 113.
- "Creating a New Virtual Service (REST)" on page 121.

- "Creating a New Connector Virtual Service (REST)" on page 145.

Defining and Managing a CloudStreams Server Target

A CloudStreams server target specifies a CloudStreams server to which you will deploy virtual services and connector virtual services. You define CloudStreams server targets using Software AG Designer. You can create one or more server targets.

To define and manage a server target

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
2. Select **Window > Preferences > Software AG > CloudStreams Servers** from the menu.
3. In the CloudStreams Servers window, click **Add**.
4. Complete the fields in the Add CloudStreams Server dialog box as follows and click **OK**.

Option	Description
Name	A name for the new target. Target names can contain alphanumeric characters and underscores (_) and hyphens (-).
Host	The server's host name (for example, `localhost`).
User	Optional. The Integration Server user who is permitted to deploy assets to this target. By default, only a member of the Integration Server's Administrator group is permitted to deploy assets to this target.
Port	The server's port number.
Password	Optional. The password of the Integration Server user who is permitted to deploy assets to this target. By default, the password of this user is `manage`.
Secure Connection	Indicates whether to open an HTTP session or an HTTPS session on the selected server. ■ If you want to open an HTTPS session on the selected server using the Secure Socket Layer (SSL), select this option. **Note:** If you select this option, it is critical that you also specify the **IS Truststore Name** option for CloudStreams (in Integration Server Administrator, go to **Solutions > CloudStreams > Administration > General**). For details, see "Setting the General Options" on page 26.

Option	Description
	■ If you want to open an HTTP session on the selected server, clear this check box.

5. In the CloudStreams Servers window, test the server connection by clicking the name of the server in the list and clicking the **Test** button. If the connection is not active and valid, activate the deployment endpoint and modify the user credentials as required.

6. To edit any parameter, click the **Edit** button, type your changes and click **OK**.

7. To delete the server from the list, click the **Remove** button.

8. To export target instances from the CloudStreams to an archive file on the file system, click the **Export** button.

9. To import target instances from the archive file into the same CloudStreams or to another instance of the server, click the **Import** button.

Creating a CloudStreams Governance Project

You need to create a CloudStreams Governance project in which you will create your virtual services, connector virtual services, and their policies. You create projects in your local file system, using the CloudStreams Development plug-in provided by Designer. You can create one or more projects. Each project will contain the folders in which you will create virtual services, connector virtual services, and their policies.

Note:
Instead of creating a new CloudStreams Governance project, you can import an existing one. To do this, from the Software AG Designer menu click **File > Import > Software AG > CloudStreams Governance Project** and complete the dialog box that appears.

To create a CloudStreams Governance project

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

   The CloudStreams Governance view on the left side of the page lists all existing CloudStreams Governance projects.

2. Right-click an existing CloudStreams Governance project and click **New > CloudStreams Governance Project** (or click **File > New > CloudStreams Governance Project** from the menu.

3. In the New CloudStreams Governance Project wizard, complete the following fields and click **Finish**.

Option	Description
Project Name	A project name that is a valid resource name on your operating system. The name must not be null, cannot be an empty string, and the following invalid OS resource characters are not allowed: ■ `\\` (double backward slashes) ■ `/` (forward slash) ■ `:` (colon) ■ `*` (asterisk) ■ `?` (question mark) ■ `"` (double quote) ■ `<` (Less Than symbol) ■ `>` (Greater Than symbol) ■ `|` (vertical bar)
Use Default Location	This option is selected by default. The default location is the Workspace root. For example, if you are using `C:\Workspaces\My_Workspace.` . the default location would be `C:\Workspaces\runtime-Eclipse Application`, which is the Workspace root. If you want to specify a different location, clear the **Use Default Location** check box, click **Browse** and select a location.
Location	Select a location in your local file system to store the new project.
Choose file system	Choose a file system, either “default” or “RSE”.
Publisher	Optional. Provide the name of the publisher of the project.
Description	Optional. A description of the project.

The new project is added to any existing projects in the CloudStreams Governance view, and it includes the default folders **Virtual Services**, **Connector Virtual Services** and **Policies**. You can create virtual services, connector virtual services and policies as described in the following sections.

4. To delete a project:

   a. Undeploy all services in the project by right-clicking the project name and clicking **Undeploy**.

b. Right-click the project name and click **Delete**.

5. To export a project, right-click the project name and click **Export** and complete the dialog box that appears.

Creating a New Virtual Service (SOAP)

To create a new virtual service (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, right-click the CloudStreams Governance project and click **New > Virtual Service** (or expand the project, right-click the **Virtual Services** folder and click **New Virtual Service**).

3. In the New Virtual Service wizard, complete the following fields and click **Finish**.

Option	Description
Project	Click **Browse** and select a CloudStreams Governance project in which to create the virtual service.
Name	Assign a name for the service. Unlike native services, the names of virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the virtual service name, then the names of the services themselves must not contain characters that are invalid in virtual service names. **Note:** If you want to change the service name after it has been created, right-click the service name in the **Virtual Services** folder and select **Rename**.
Version	The version is always set to 1.0.
Type	Select **SOAP**.
WSDL	The WSDL that the virtual service uses. Select either **File** (and click **Browse** to select a WSDL) or select **URL** (and enter the URL of the WSDL). **Note:** If you need to add a WSDL to the service later, you can leave the WSDL field blank and add the WSDL later (see "Managing a Virtual Service (SOAP)" on page 65.

Option	Description
Description	Optional. A description for the virtual service. This description appears when a user displays a list of virtual services in the user interface.

The new virtual service is added to the CloudStreams Governance project, in the **Virtual Services** folder.

The Properties of a Virtual Service (SOAP)

To view and modify the properties of a virtual service (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**. .
2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name. (If the Properties view is not already open, click **Window > Show View > Other > General > Properties**.)
3. The **General** page in the Properties view displays the following properties:

Option	Description
Name	(Read-only field.) The service name. You can change the name of an undeployed service by right-clicking the name in the **Virtual Services** folder and clicking **Rename**.
Service Type	(Read-only field.) **SOAP**.
Created/Last Modified	(Read-only field.) The service's creation/modification timestamps.
Target Namespace	(Read-only field.) This value is derived from the `targetNamespace` attributes of the WSDL's definition element.
Namespaces	Click the button next to this field to view the virtual service's available namespaces (such as `wsdl`, `xsd`, `soap`, etc.).
Version	The version is always set to 1.0.
WSDL URL	Click this URL to display the contents of the service's abstract WSDL. If a WSDL file was not added, this will be empty. You can override the WSDL by attaching a new one; see "Managing a Virtual Service (SOAP)" on page 65.
Description	You can change the service description.

4. View additional properties in the **Advanced** page.

Option	Description
Name	(Read-only field.) The service name. You can change the name of an undeployed service by right-clicking the name in the **Virtual Services** folder and clicking **Rename**.
Type	(Read-only field.) **SOAP**.
Target Namespace	(Read-only field.) The value derived from the `targetNamespace` attributes of the WSDL's definition element.
WSDL URL	Click this URL to display the contents of the service's abstract WSDL. If a WSDL file was not added, this will be empty. You can override the WSDL by attaching a new one; see "Managing a Virtual Service (SOAP)" on page 65.
Namespaces	Click the button next to this field to view the virtual service's available namespaces (such as `wsdl`, `tns`, `xsd`, `soap`, and so on).
Version	The version is always set to 1.0.
Description	(Read-only field.) The service description.
VSD	Click the button next to this field to view the service's "virtual service definition" (VSD). This button is disabled until you deploy the virtual service to a CloudStreams server. When you deploy the virtual service to a CloudStreams server, CloudStreams generates an XML document called a *virtual service definition (VSD)*. The VSD defines the virtual service for CloudStreams, and contains all the resources required to deploy the virtual service to a CloudStreams server, including the policy that applies to the service. You cannot edit the VSD. **Note:** If multiple policies apply to the service, CloudStreams combines all those policies into a single policy known as the *effective policy*. The effective policy is a simple UNION of the run-time actions specified in all policies that apply to a service. To create the effective policy, CloudStreams evaluates the combined list of actions from all policies, using a set of internal rules known as Policy Conflict Resolution rules. For details, see "Policy Conflict Resolution Rules" on page 195.
Applicable Policies	Click the button next to this field to view a list of the active policy or policies that apply to this service. Any inactive policy that applies to the service is not listed. To change the list of applicable policies, see "Modifying Policies" on page 164.

Option	Description
Endpoint	Click the button next to this field to view the endpoint to which the virtual service is deployed, if applicable.
Deployed Status	(Read-only field.) Indicates whether the service is **Deployed**, **Undeployed** or **Not Deployed** (which is the initial status before you deploy the service). To deploy or undeploy a service, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Resource	Not applicable to SOAP services.

Managing a Virtual Service (SOAP)

You can rename, delete, deploy or undeploy a virtual service, attach a WSDL to it, and change the list of policies that apply to the virtual service.

To manage a SOAP virtual service

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**. .
2. In the CloudStreams Governance view, expand your CloudStreams Governance project.
3. Right-click the virtual service name and choose one of the following tasks from the context menu:

Option	Description
Rename	Use this option to assign a new name to the service. The service must be undeployed before you can rename it (check the **Deployed Status** field in the Advanced pageUse of the service's Properties view). Unlike native services, the names of virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the virtual service name, then the names of the services themselves must not contain characters that are invalid in virtual service names.
Delete	Deletes the virtual service from CloudStreams.
Attach WSDL	Use this option to attach a WSDL to an existing virtual service. The new attached WSDL will override the old WSDL. In the dialog box that appears, choose one of the following options and click **OK**:

Option	Description
	■ **URL**: Specify the URL of the WSDL to attach. ■ **File**: Click **Browse** to select a WSDL from your local file system. **Note:** If the new WSDL contains referenced XSDs or WSDL, they will be copied if they are resolved by CloudStreams. But if the referenced XSDs or WSDL cannot be resolved, a dialog box will prompt you whether to import the unresolved XSDs or WSDL.
Deploy	Deploys the virtual service to a CloudStreams server target. For more information, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Undeploy	Undeploys the virtual service from a CloudStreams server target.

4. To change the list of policies that apply to the virtual service, see "Modifying Policies" on page 164.

The "In Sequence" Step (SOAP)

You configure the service's "In Sequence" step to manipulate the request messages. This step can include the following sub-steps:

- The "Entry Step" (required), which specifies the protocol (HTTP or HTTPS) and SOAP format (1.1 or 1.2) of the requests that the virtual service will accept.
- The "Transform" step (optional), which performs an XSLT message transformation on the request message before it is submitted to the native service.
- The "Invoke IS Service" step (optional), which pre-processes the request message before it is submitted to the native service.
- The "Routing Rule" step (required), which specifies how the virtual service will route the service requests to the native service endpoint. There are four ways to route HTTP or HTTPS requests:
  - "Straight Through" routing (to route requests directly to the native service endpoint).
  - "Context-Based" routing (to route specific types of messages to specific endpoints according to context-based routing rules).
  - "Content-Based" routing (to route specific types of messages to specific endpoints based on specific values that appear in the request message).
  - "Load Balancing" routing (to distribute requests among multiple endpoints).

The "Entry Step" (SOAP)

The Entry Step (required) specifies the protocol (HTTP or HTTPS) and SOAP format (1.1 or 1.2) of the requests that the virtual service will accept. This step is required.

This step allows you to bridge protocols between the consuming application and the native service. For example, suppose you have a native service that is exposed over HTTPS and a consuming application that submits SOAP requests over HTTP. In this situation, you can configure the virtual service's Entry Step to accept HTTP requests and configure its Routing Rule step to route the request to the Web service using HTTPS.

To configure the "Entry Step" (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**. .

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

   The virtual service editor displays the three main processing steps: **In Sequence**, **Out Sequence** and **Error Sequence**.

3. In the virtual service editor, expand **In Sequence**.

   By default, the **In Sequence** step contains the sub-steps **Entry Step** and **Routing Rule**. (You can add the optional **Transform** and **Invoke IS Service** steps, as described later.)

4. Click **Entry Step** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read only.) **Entry Step**.
Protocol	The protocol (HTTP or HTTPS) over which the virtual service will accept requests. To specify HTTPS, select both **HTTP** and **SSL**.
Format	The SOAP format (SOAP 1.1 or SOAP 1.2) of the requests that the virtual service will accept.

The "Transform" Step (Inbound, SOAP)

The optional "Transform" step specifies how the SOAP request message is to be transformed before it is submitted to the native service.

No message transformation is required as long as the request message structure matches the request message structure that is required by the operation associated with the `soapAction`. However, in some cases a virtual service might need to transform SOAP messages.

For example, you might need to accommodate differences between the message content that a consuming application is capable of submitting and the message content that a native service expects. For example, if the consuming application submits an order record using a slightly different structure than the structure expected by the native service, you can use the Transform step to transform the record submitted by the consuming application to the structure required by the Web service.

In this case, you would need to create two Transform steps:

- One in the "In Sequence" step, to transform the request messages into the format required by the native service, before CloudStreams sends the requests to the native services. To do this, you pass the message to an XSLT transformation file. (Additionally in this case, the transformation is required if the virtual service has a schema validation policy that validates the requests.)
- One in the "Out Sequence" step, to transform the native service's response messages into the format required by the consumer applications, before CloudStreams returns the responses to the consumer applications.

To add the "Transform" step (inbound, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.
3. Right-click **In Sequence** and select **Transform**.

   The Transform step is added under the Entry Step. You cannot change the order of the steps.
4. Click **Transform** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only.) **Transform**.
XSLT File	The XSLT file to transform the request message before it is submitted to the native service. Click **Browse** to select a file from your file system and click **Save**. The XSL file uploaded by the user should not contain the XML declaration in it (`xml version="1.0" encoding="UTF-8"`). This is

Option	Description
	because when the virtual service is deployed to CloudStreams, CloudStreams embeds the XSL file in the virtual service definition (VSD), and since the VSD itself is in XML format, there cannot be an XML declaration line in the middle of it. This can lead to unexpected deployment issues which can be avoided by making sure the XSL file does not contain the declaration line. **Note:** If you make changes to the XSLT file in the future, you must re-deploy the virtual service.

5. To create an additional Transform step, right-click **In Sequence** and select **Transform** again.

6. To delete a Transform step, right-click **Transform** and click **Delete Step**.

The "Invoke IS Service" Step (Inbound, SOAP)

This optional step invokes a user-defined Integration Server flow service. You can invoke an IS service in:

- The "In Sequence" step, to pre-process the request message before it is submitted to the native service.
- The "Out Sequence" step, to pre-process the response message from the native service before it is returned to the consuming application.

The IS service you invoke must be running on the same Integration Server as CloudStreams. It can call out a C++ or Java or .NET function. It can also call other IS services to manipulate the SOAP message.

You can use the following constructs in an IS service:

- Predefined or custom context variables. For more information, see "Using Context Variables in IS Services" on page 96.
- The Security API provided by CloudStreams (for SOAP-based services only). For more information, see "Using the Security API in IS Services" on page 88.

You can create multiple "Invoke IS Service" steps.

To add the "Invoke IS Service" step (inbound, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Right-click **In Sequence** and click **Invoke IS Service**.

   The "Invoke IS Service" step is added under the Entry Step (and under a Transform step, if one exists).

4. Click **Invoke IS Service** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only.) **Invoke IS Service**.
Service	The IS service to pre-process the request message before it is submitted to the native service.

5. When you define the IS service, the **Pipeline In** section of the flow should have the following input variables:

   - `proxy.name`: This is the name of the virtual service.
   - `SOAPEnvelope`: This is of the Java type org.apache.axiom.soap.SOAPEnvelope.
   - `MessageContext`: CloudStreams will automatically place a `MessageContext` variable into the pipeline before executing the IS service call. `MessageContext` is of the Java type org.apache.axis2.context.MessageContext.

     Integration Server users can use the Axis2 `MessageContext` object to manipulate the incoming SOAP request. The Integration Server provides built-in services (the pub.soap.* services) to work with the `MessageContext` object to get/set/modify the SOAP body, header, properties, etc. Integration Server users should use these services to extract the information they need from the `MessageContext` to build the necessary business logic. Users do not need to understand Axis2 or Axiom (the XML object model based on StAX) to work with the SOAP request, because if they are familiar with the Integration Server pub.soap.* services, they can accomplish most of the tasks. For more information about these related Integration Server services, see the *webMethods Integration Server Built-In Services Reference*.

6. To create an additional Invoke IS Service step, right-click **In Sequence** and select **IS Service** from the context menu.

7. To delete an Invoke IS Service step, right-click **Invoke IS Service** and click **Delete**.

The "Routing Rule" Step for HTTP or HTTPS (SOAP)

The Routing Rule step is required. You can choose one of the following protocols in your Routing Rule step for routing the requests:

- "Straight Through" routing (to route requests directly to the native service endpoint).

- "Context-Based" routing (to route specific types of messages to specific endpoints according to context-based routing rules).
- "Content-Based" routing (to route specific types of messages to specific endpoints based on specific values that appear in the request message).
- "Load Balancing" routing (to distribute requests among multiple endpoints).

The "Straight Through" Routing Rule Step (SOAP)

When you select the "Straight Through" routing protocol, the virtual service will route the requests directly to the native service endpoint you specify. You may specify how to authenticate requests (as with all routing protocols).

To configure the "Routing Rule" step for "Straight Through" routing (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.
3. Expand **In Sequence**.
4. Click **Routing Rule** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Straight Through**.

Default To	Specify the URL of the service to which to route the request.
	
	Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows:
	
	■ **SOAP Optimization Method**: Select one of the following options:
	
	  ■ **None**: The default.
	
	  ■ **MTOM**: Indicates that CloudStreams expects to receive a request with a Message Transmission Optimization

Option	Description

Mechanism (MTOM) attachment, and will forward the attachment to the native service.

- **SwA**: Indicates that CloudStreams expects to receive a "SOAP with Attachment" (SwA) request, and will forward the attachment to the native service.

  Note:
  Bridging between SwA and MTOM is not supported. If a consumer sends an SwA request, CloudStreams can only forward SwA to the native provider. The same is true for MTOM, and applies to responses received from the native provider. That is, an SwA or MTOM response received by CloudStreams from a native provider will be forwarded to the caller using the same format it received.

  Note:
  When sending SOAP requests that do not contain a MTOM or SWA attachment to a virtual service for a native provider endpoint that returns an MTOM or SWA response, the request `Accept` header must be set to 'multipart/related' (or the virtual service's Request Processing Step should include an IS service callout that sets the BUILDER_TYPE context variable to 'multipart/related'). This is necessary so CloudStreams knows how to parse the response properly.

- **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server.
- **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds.
- **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank.
  - **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication.
  - **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This

Option	Description
	value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.
Use credentials from incoming request	Default. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use specific credentials	Authenticates requests according to the values you specify in the **User** and **Password** fields, and optionally the **Domain** field.
Invoke service anonymously	Does not authenticate requests.
Use existing HTTP Headers	Use the HTTP headers that are contained in the requests.
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on this page. If you need to specify multiple headers, use the plus button to add rows.

The "Context-Based" Routing Rule Step (SOAP)

If you have a native service that is hosted at two or more endpoints, you can use the Context-Based routing protocol to route specific types of messages to specific endpoints according to the context-based routing rules you create.

A routing rule specifies where the requests should be routed, and the criteria by which they should be routed there. For example, requests can be routed according to certain consumers, certain dates/times, or according to requests that exceed/fall below a specified metric (Total Count, Success Count, Fault Count, etc.). You can create one or more rules. For example, you might use this capability to route requests from certain high-priority consumers to endpoints on a fast machine.

To configure the "Routing Rule" step for Context-Based routing (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Context-Based**.
Rule Name	Assign a name to the rule. Then, click the icon next to this field and choose one of the following variables in the Configure Routing Rule dialog box that appears: **Time**, **IP Address** (IPv4 or IPv6 format), **Date**, **Consumer**, **Predefined Context Variable** or **Custom Context Variable** (see "Using Context Variables in IS Services" on page 96). Then specify a value and operator appropriate for your chosen variable. If you need to specify multiple variables, use the plus button at the end of the row to add rows. **Note:** If you select the value **Custom Context Variable**, you must write an IS service to get/set the custom context variable, and then invoke that service in an **Invoke IS Service** step. CloudStreams provides an API to get/set custom context variables. For more information, see "The API for Context Variables" on page 100. CloudStreams automatically declares any custom context variables you have specified; there is no need for you to declare them.
Route To	Specify the URL of the native service to route the request to, if the rule criteria are met. Click the icon next to this field and complete the **Configure Endpoint Properties** dialog box as follows: ■ **Optimization Method**: Select one of the following options: ■ **None**: The default. ■ **MTOM**: Indicates that CloudStreams expects to receive a request with a Message Transmission Optimization Mechanism (MTOM) attachment, and will forward the attachment to the native service. ■ **SwA**: Indicates that CloudStreams expects to receive a "SOAP with Attachment" (SwA) request, and will forward the attachment to the native service.

Option	Description

Note:
Bridging between SwA and MTOM is not supported. If a consumer sends an SwA request, CloudStreams can only forward SwA to the native provider. The same is true for MTOM, and applies to responses received from the native provider. That is, an SwA or MTOM response received by CloudStreams from a native provider will be forwarded to the caller using the same format it received.

Note:
When sending SOAP requests that do not contain a MTOM or SWA attachment to a virtual service for a native provider endpoint that returns an MTOM or SWA response, the request `Accept` header must be set to `multipart/related` (or the virtual service's "In Sequence" step should include an IS service callout that sets the `BUILDER_TYPE` context variable to `multipart/related`). This is necessary so CloudStreams knows how to parse the response properly.

- **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server.
- **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds.
- **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank.
  - **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication.
  - **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.

Default To	Enter a native service endpoint to route the request to in case all routing rules evaluate to False. Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box, as described for the **Route To** field above.

Option	Description
Use credentials from incoming request	Default. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use specific credentials	Authenticates requests according to the values you specify in the **User**, **Password** and **Domain** fields.
Invoke service anonymously	Does not authenticate requests.
Use existing HTTP Headers	Use the HTTP headers that are contained in the requests.
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on this page. If you need to specify multiple headers, use the plus button to add rows.

The "Content-Based" Routing Rule Step (SOAP)

If you have a native service that is hosted at two or more endpoints, you can use the Content-Based routing protocol to route specific types of messages to specific endpoints based on specific values that appear in the request message.

You might use this capability, for example, to determine which operation the consuming application has requested, and route requests for complex operations to an endpoint on a fast machine.

The requests are routed according to the content-based routing rules you create. That is, they are routed based on the successful evaluation of one or more XPath expressions that are constructed utilizing the content of the request payload. For example, a routing rule might allow requests for half of the methods of a particular service to be routed to Service A, and the remaining methods to be routed to Service B.

To configure the "Routing Rule" step for Content-Based routing (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Content-Based**.
Rule Name	Assign a name to the rule.
XPath	Create an XPath expression as follows: a. Click the icon next to this field to display the XPath Editor. b. In the XPath Editor that appears, the **Namespace** tab displays all predefined namespaces in the WSDL. If you want to add custom namespaces, click **Add Custom Namespace/prefix**, specify a name and value for the namespace, and click **OK**. To add additional rows, use the plus button at the end of the row to add them. c. In the XPath Editor's **Nodes** tab, expand the namespace's node, choose the method you want for the XPath expression, and click **OK**. d. In the XPath Editor's **Evaluator** tab, evaluate the XPath expression by specifying an argument in the **XPath Expression** field, and clicking **Evaluate**. The true/false result of the evaluation is displayed in the **Result** field. To specify additional XPath expressions, use the plus button at the end of the row to add them.
Route To	Specify where to route the request if the rule criteria are met. Specify either the URL of a native service or a connection pool name. Click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows: ■ **Optimization Method**: Select one of the following options: ■ **None**: The default. ■ **MTOM**: Indicates that CloudStreams expects to receive a request with a Message Transmission Optimization Mechanism (MTOM) attachment, and will forward the attachment to the native service.

Option	Description

- **SwA**: Indicates that CloudStreams expects to receive a "SOAP with Attachment" (SwA) request, and will forward the attachment to the native service.

  Note:
  Bridging between SwA and MTOM is not supported. If a consumer sends an SwA request, CloudStreams can only forward SwA to the native provider. The same is true for MTOM, and applies to responses received from the native provider. That is, an SwA or MTOM response received by CloudStreams from a native provider will be forwarded to the caller using the same format it received.

  Note:
  When sending SOAP requests that do not contain a MTOM or SWA attachment to a virtual service for a native provider endpoint that returns an MTOM or SWA response, the request `Accept` header must be set to 'multipart/related' (or the virtual service's Request Processing Step should include an IS service callout that sets the BUILDER_TYPE context variable to 'multipart/related'). This is necessary so CloudStreams knows how to parse the response properly.

- **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server.
- **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds.
- **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank.
  - **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication.
  - **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.

Option	Description
Default To	Enter a native service endpoint to route the request to in case all routing rules evaluate to False. Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box, as described for the **Route To** field above.

The "Load Balancing" Routing Rule Step (SOAP)

If you have a native service that is hosted at two or more endpoints, you can use the Load Balancing protocol to distribute requests among the endpoints.

The requests are intelligently routed based on the "round-robin" execution strategy. The load for a service is balanced by directing requests to two or more services in a pool, until the optimum level is achieved. The application routes requests to services in the pool sequentially, starting from the first to the last service without considering the individual performance of the services. After the requests have been forwarded to all the services in the pool, the first service is chosen for the next loop of forwarding.

Load-balanced endpoints also have automatic Failover capability. If a load-balanced endpoint is unavailable (for example, if a connection is refused), then that endpoint is marked as "down" for the number of seconds you specify in the **Suspend Interval** field (during which the endpoint will not be used for sending the request), and the next configured endpoint is tried. If all the configured load-balanced endpoints are down, then a SOAP fault is sent back to the client. After the suspension period expires, each endpoint marked will be available again to send the request.

To configure the "Routing Rule" step for "Load Balancing" routing (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.

Option	Description
Routing Type	Select **Load Balancing**.

Route To	Specify the URLs of two or more services in a pool to which the requests will be routed. The application routes the requests to services in the pool sequentially, starting from the first to the last service without considering the individual performance of the services. After the requests have been forwarded to all the services in the pool, the first service is chosen for the next loop of forwarding.
	
	To specify additional URLs, use the plus button next to the field to add rows.
	
	Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows. These properties will apply to all the endpoints.
	
	■ **Optimization Method**: Select one of the following options:
	  ■ **None**: The default.
	  ■ **MTOM**: Indicates that CloudStreams expects to receive a request with a Message Transmission Optimization Mechanism (MTOM) attachment, and will forward the attachment to the native service.
	  ■ **SwA**: Indicates that CloudStreams expects to receive a "SOAP with Attachment" (SwA) request, and will forward the attachment to the native service.
	
	Note:
	Bridging between SwA and MTOM is not supported. If a consumer sends an SwA request, CloudStreams can only forward SwA to the native provider. The same is true for MTOM, and applies to responses received from the native provider. That is, an SwA or MTOM response received by CloudStreams from a native provider will be forwarded to the caller using the same format it received.
	
	Note:
	When sending SOAP requests that do not contain a MTOM or SWA attachment to a virtual service for a native provider endpoint that returns an MTOM or SWA response, the request `Accept` header must be set to 'multipart/related' (or the virtual service's Request Processing Step should include an IS service callout that sets the BUILDER_TYPE context variable to 'multipart/related'). This is necessary so CloudStreams knows how to parse the response properly.

Option	Description

	■ **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server.
	■ **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds.
	■ **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank.
	  ■ **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication.
	  ■ **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.

Suspend Interval	A numeric timeout value (in seconds) for a failed endpoint. Default: 30. When this timeout value expires, the system routes the execution of the virtual service to the next consecutive Web service endpoint specified in the **Route To** field.
Use Credentials from Incoming Request	Default. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use Specific Credentials	Authenticates requests according to the values you specify in the **User**, **Password** and **Domain** fields.
Invoke Service Anonymously	Does not authenticate requests.
Use Existing HTTP Headers	Use the HTTP headers that are contained in the requests.
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on this page. If you need to specify multiple headers, use the plus button to add rows.

The "Out Sequence" Step (SOAP)

You configure the service's "Out Sequence" step (optional) to manipulate the response messages from the native service before they are returned to the consuming applications. This step includes the following sub-steps:

- The "Transform" step (optional), which invokes an XSLT transformation file to transform the SOAP response payloads from XML format to the format required by the consumer.
- The "Invoke IS Service" step (optional), which processes the response message from the native service provider before it is returned to the consuming application.

The "Transform" Step (Outbound, SOAP)

Add this step if you need to invoke an XSLT transformation file to transform the native service's response messages into the format required by the consumer applications, before CloudStreams returns the responses to the consumer applications. For more information about when to use the Transform step, see "The Transform Step (Inbound, SOAP)" on page 67.

To add the "Transform" step (outbound, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Right-click **Out Sequence** and click **Transform**.

   The Transform step is added to the Out Sequence step.

4. Click **Transform** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Transform**.
XSLT File	The XSLT file to transform the response message before it is returned to the consuming application. Click **Browse** to select a file from your file system and click **Save**. The XSL file uploaded by the user should not contain the XML declaration in it (`xml version="1.0" encoding="UTF-8"`). This is because when the virtual service is deployed to CloudStreams, CloudStreams embeds the XSL file in the virtual service definition

Option	Description
	(VSD), and since the VSD itself is in XML format, there cannot be an XML declaration line in the middle of it. This can lead to unexpected deployment issues which can be avoided by making sure the XSL file does not contain the declaration line. **Note:** If you make changes to the XSLT file in the future, you must re-deploy the virtual service.

5. To create an additional Transform step, right-click **In Sequence** and select **Transform** again.

6. To delete a Transform step, right-click **Transform** and click **Delete**.

The "Invoke IS Service" Step (Outbound, SOAP)

This optional step invokes a user-defined Integration Server flow service. You can invoke an IS service in the service's "Out Sequence" step to pre-process the response message from the native service before it is returned to the consuming application. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69.

To add the Invoke IS Service step (outbound, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

3. Right-click **Out Sequence** and click **Invoke IS Service**.

   The Invoke IS Service step is added to the Out Sequence step.

4. Click **Invoke IS Service** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Invoke IS Service**.
Service	The IS service to pre-process the response message from the native service before it is returned to the consuming application.

5. When you define the IS service, the **Pipeline In** section of the flow should have the following input variables:

   - `proxy.name`: This is the name of the virtual service.
   - `SOAPEnvelope`: This is of the Java type org.apache.axiom.soap.SOAPEnvelope.
   - `MessageContext`: CloudStreams will automatically place a `MessageContext` variable into the pipeline before executing the IS service call. `MessageContext` is of the Java type org.apache.axis2.context.MessageContext.

     Integration Server users can use the Axis2 `MessageContext` object to manipulate the incoming SOAP request. The Integration Server provides built-in services (the pub.soap.* services) to work with the `MessageContext` object to get/set/modify the SOAP body, header, properties, etc. Integration Server users should use these services to extract the information they need from the `MessageContext` to build the necessary business logic. Users do not need to understand Axis2 or Axiom (the xml object model based on StAX) to work with the SOAP request, because if they are familiar with the Integration Server pub.soap.* services, they can accomplish most of the tasks. For more information about these related Integration Server services, see the *webMethods Integration Server Built-In Services Reference*.

6. To create an additional Invoke IS Service step, right-click **In Sequence** and click **IS Service** again.

7. To delete an Invoke IS Service step, right-click **Invoke IS Service** and click **Delete**.

The "Error Sequence" Step (SOAP)

CloudStreams returns a default fault response to the consuming application, which you can customize with context variables. This fault response is used for faults returned by the native service provider as well as faults returned by internal CloudStreams exceptions (policy violation errors, cloud connection errors and cloud connector service errors). In addition, you can:

- Choose whether or not to send the native service provider's service fault content, or just send the response message.
- Invoke IS services to pre-process or post-process the error messages.

You use this step to configure error messaging for each virtual service individually. If you want to configure global error responses for *all* virtual services, set the Service Fault Configuration options, which are located in the Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Service Fault Configuration**). For details, see "Setting the Service Fault Configuration Options" on page 39.

To configure the Error Sequence step (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

3. Expand **Error Sequence**.

4. Click **Error Messaging** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Error Messaging**.
Error Message	Select one or both of the following options: **Custom Failure Response Message**: When you select this option, CloudStreams returns the following fault response to the consuming application: `CloudStreams encountered an error:$ERROR_MESSAGE while executing operation:$OPERATION service:$SERVICE at time:$TIME on date:$DATE. The client ip was:$CLIENT_IP. The current user:$USER. The consumer application:$CONSUMER_APPLICATION".` **Note:** When $CLIENT_IP is used, CloudStreams will replace $CLIENT_IP with the IP address of the client. For privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), delete the *"The client ip was:$CLIENT_IP."* string. This fault response is returned in both of the following cases: ■ When a fault is returned by the native service provider. In this case, the `$ERROR_MESSAGE` variable in the fault response will contain the message produced by the provider's exception that caused the error. This is equivalent to the `getMessage` call on the Java Exception. This maps to the faultString element for SOAP 1.1 or the Reason element for SOAP 1.2 catch. CloudStreams discards the native service provider's fault and does not return this content to the web service caller since it could be considered a security issue, especially if the native provider is returning a stack trace with its response. ■ When a fault is returned by internal CloudStreams exceptions (policy violation errors, cloud connection errors and cloud connector service errors).

Option	Description
	In this case, the `$ERROR_MESSAGE` variable will contain the error messages generated by CloudStreams. The default fault response contains predefined fault handler variables (`$ERROR_MESSAGE`, `$OPERATION`, etc.), which are described in "The Fault Handler Variables" on page 87. You can customize the default fault response using the following substitution variables, where CloudStreams replaces the variable reference with the real content at run time: ■ The predefined context variables listed in "The Predefined Context Variables" on page 97. ■ Custom context variables that you declare using the CloudStreams API (see "The API for Context Variables" on page 100). **Note:** If you want to reference a custom context variable that you have already defined in a context-based routing rule (as opposed to one you have declared using CloudStreams API for context variables), then you must add the prefix `$mx` to the variable name in order to reference the variable. For example, if you defined the variable `TAXID`, you would reference it as `$mx:TAXID`. **Native Provider Fault**: When you select this option, CloudStreams sends the native service provider's service fault content, if one is available. CloudStreams will send whatever content it received from the native service provider. If you select this option, the **Custom Failure Response Message** is ignored when a fault is returned by the native service provider. (Faults returned by internal CloudStreams exceptions will still be handled by the **Custom Failure Response Message** option.)
Processing Method	Optionally select either of the following: ■ **Pre-Processing**: Select this option if you want to invoke an IS service to manipulate the response message before the Error Sequence step is invoked. The IS service will have access to the response message context (the axis2 `MessageContext` instance) before it is updated with the custom error message. For example, you might want to send emails or perform custom alerts based on the response payload. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69. ■ **Post-Processing**: Select this option if you want to invoke an IS service to manipulate the service fault after the Error Sequence

Option	Description
	step is invoked. The IS service will have access to the entire service fault and the custom error message. You can make further changes to the fault message structure, if needed. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69.

The Fault Handler Variables

The following fault handler variables are used in fault response messages. You can define fault response messages either in a virtual service's "Error Sequence" step or in the global Service Fault Configuration page in Integration Server Administrator.

Error Handler Variable	Description
`$ERROR_MESSAGE`	When a fault is returned by the native service provider, this variable will contain the message produced by the provider's exception that caused the error. This is equivalent to the `getMessage` call on the Java Exception. This maps to the faultString element for SOAP 1.1 or the Reason element for SOAP 1.2 catch. For REST service calls, the message is returned inside an `</Exception>` tag. If the response is XML, the message is returned inside `<Exception>'custom message'</Exception>`. If the response is JSON, it will be returned inside `{"Exception":"Invalid response"}`. When a fault is returned by internal CloudStreams exceptions (policy violation errors, cloud connection errors and cloud connector service errors), this variable will contain the error messages generated by CloudStreams.
`$OPERATION`	The operation that was invoked when this error occurred.
`$SERVICE`	The service that was invoked when this error occurred. **Note:** Deprecated, but still a valid alias. Alternatively, use `$SERVICE_NAME` (see "The Predefined Context Variables" on page 97).
`$TIME`	The time (as determined on the Container side) at which the error occurred.
`$DATE`	The date (as determined on the Container side) at which the error occurred.
`$CLIENT_IP`	The IP address of the client invoking the service. This might be available for only certain invoking protocols, such as HTTP(S).

Error Handler Variable	Description
	Note: Deprecated, but still a valid alias. Alternatively, use `$INBOUND_IP` (see "The Predefined Context Variables" on page 97).
`$USER`	The currently authenticated user. The user will be present only if the Transport/SOAP Message have user credentials.
`$CONSUMER_APPLICATION`	The currently identified consumer application. If the service's policy does not contain the "Identify Consumer" action, `$CONSUMER_APPLICATION` will return "null".

IS Service Constructs

As previously mentioned, an IS (Integration Server) service is a user-defined Integration Server flow service that you can invoke in the "In Sequence" and "Out Sequence" steps of virtual services (both SOAP-based and REST-based).

You can use the following constructs in an IS service:

- The Security API provided by CloudStreams (for SOAP-based services only). For more information, see "Using the Security API in IS Services" on page 88.
- Predefined or custom context variables. For more information, see "Using Context Variables in IS Services" on page 96.

Using the Security API in IS Services

Note:
This API can only be used for SOAP-based virtual services.

CloudStreams provides Java services that you can use to support WS-Security functionality in an IS service that you invoke in the "In Sequence" step. These following services are included:

- "pub.cloudstreams.security.ws.AddUserNameToken" on page 88
- "pub.cloudstreams.security.ws.AddX509Token" on page 91
- "pub.cloudstreams.security.ws.AddSamlSenderVouchesToken" on page 92
- "pub.cloudstreams.security.ws.AddTimestamp" on page 93
- "pub.cloudstreams.security.ws.AddWSAddressingHeaders" on page 94

pub.cloudstreams.security.ws.AddUserNameToken

Adds the WS-Username Token 1.0 and 1.1 to the request. This service includes the following input parameters:

Parameter	Data Type	Description	Default Value
username	String	Required. Java Type: String. The value that will be added as the Username element in the token.	""
MessageContext	Object	Required. Java Type: org.apache.axis2. context. MessageContext. CloudStreams will place a MessageContext parameter into the pipeline before executing the IS service call.	org.apache. axis2.context. MessageContext instance
password	String	Java Type: String. The password for the token; must be specified if the passwordType (see below) is specified as either TEXT or DIGEST.	""
passwordType	String	Java Type: String. Specifies how the password will be added in the token. Valid values: ■ NONE: The password is not added. ■ TEXT: The password is added in plain text. ■ DIGEST: The password is added in digested form (as specified in the UsernameToken profile).	NONE
addNonce	Boolean	Java Type: Boolean. Specifies whether the Nonce element will be added to the token.	False
addCreated	Boolean	Java Type: Boolean. Specifies whether the Created element will be added to the token.	False
salt	byte[]	Java Type: byte[]. The value for the /wsse11:UsernameToken/wsse:Salt element. Its value is a 128 bit	null

Parameter	Data Type	Description	Default Value
		number serialized as `xs:base64Binary`.	
`iteration`	int	Java Type: Integer. Indicates the number of times the hashing operation is repeated when deriving the key. It is expressed as a `xs:unsignedInteger` value. If it is not present, a value of `1000` is used for the iteration count.	`1000`
`useMac`	Boolean	Java Type: Boolean. Indicates whether the derived key will be used as a Message Authentication Code (MAC) or as a symmetric key for encryption.	`False`
`useBasicAuthCredentials`	Boolean	Java Type: Boolean. If this parameter is set to `True`, CloudStreams will try to use the username and password from the `Authorization` HTTP header. In this case the `username` and `password` fields need not be specified.	`False`
`actor`	String	Java Type: String. Indicates the value of the SOAP `actor` attribute if a new security header is being added to the SOAP request. If the request already has a security header with the `actor` specified in it, then this value will not overwrite it.	`""`
`mustUnderstand`	Boolean	Java Type. Boolean. Specifies whether the security header will have the `mustUnderstand` attribute set to 0 or 1 (false / true). If the security header already has this attribute set, then this value will not overwrite it.	`False`

pub.cloudstreams.security.ws.AddX509Token

Adds an X.509 certificate (or certificate chain) as a `BinarySecurityToken` (BST) element in the outbound SOAP request. This service includes the following input parameters:

Parameter	Data Type	Description	Default Value
`MessageContext`	Object	Required. Java Type: `org.apache.axis2.context.MessageContext`. CloudStreams will place a `MessageContext` parameter into the pipeline before executing the IS service call.	`org.apache.axis2.context.MessageContext` instance
`keystoreFile`	String	Required. Java Type: String. The absolute path to a keystore file on the system where CloudStreams is running.	`""`
`keystorePassword`	String	Java Type: String. Required. The password for the keystore.	`""`
`keystoreType`	String	Java Type: String. The type of keystore represented by the file (can be `JKS`, `JCEKS` or `PKCS12`).	`JKS`
`keyAlias`	String	Java Type: String. Required. The key alias whose X509 certificate will be sent in the SOAP request as a `BST`.	`""`
`useCertificatePath`	Boolean	Java Type: Boolean. If set to `True`, CloudStreams will use the entire certificate chain represented by the key alias instead of just a single certificate.	`False`
`actor`	String	Java Type: String. Indicates the value of the SOAP `actor` attribute if a new security header is being added to the SOAP request. If the request already has a security header with this attribute specified in it, then this value will not overwrite it.	`""`
`mustUnderstand`	Boolean	Java Type: Boolean. Specifies whether the security header will have the `mustUnderstand` attribute set to 0 or 1 (false / true). If the	`False`

Parameter	Data Type	Description	Default Value
		security header already has this attribute set, then this value will not overwrite it.	

pub.cloudstreams.security.ws.AddSamlSenderVouchesToken

This service enables a Security Token Service (STS) client to send a WS-Trust request to a configured STS to obtain a SAML v1/v2 assertion. For the details about configuring CloudStreams to act as an STS client, see "Setting the STS Options" on page 35.

This service adds the obtained SAML assertion to the original request that is sent by the client to the native service, and includes the following parameters.

Parameter	Data Type	Description	Default Value
`ConfigName`	String	Required. Java Type: String. References a previously configured STS configuration name.	`""`
`MessageContext`	Object	Required. Java Type: `org.apache.axis2.context.MessageContext`. CloudStreams will place a `MessageContext` parameter into the pipeline before executing the IS service call.	`org.apache.axis2.context.MessageContext` instance
`addTimeStamp`	Boolean	Java Type: Boolean. Adds a `Timestamp` element (with the duration specified in `timeToLive`) to the WS-Security header of the request, and includes it in the signature. (The other items that are signed are the body and SAML assertion.)	`False`
`timeToLive`	Integer	Java Type: Integer. If `addTimeStamp` is true, then `timeToLive` specifies the duration (in seconds) for which the request is valid.	`300` seconds (5 minutes)
`actor`	String	Java Type: String. Indicates the value of the SOAP `actor` attribute if a new security header is being added to the SOAP request. If the request already has a security header with this attribute specified in it, then this value will not overwrite it.	`""`

Parameter	Data Type	Description	Default Value
`mustUnderstand`	Boolean	Java Type: Boolean. Specifies whether the security header will have the `mustUnderstand` attribute set to 0 or 1 (false / true). If the security header already has this attribute set, then this value will not overwrite it.	`False`

pub.cloudstreams.security.ws.AddTimestamp

Adds a timestamp to the outbound SOAP request WS-Security header. This service includes the following input parameters:

Parameter	Data Type	Description	Default Value
`timeToLive`	Integer	Java Type: Integer. Specifies the duration (in seconds) for which the request is valid.	`300` seconds (5 minutes)
`signTimestamp`	Boolean	Java Type: Boolean. Indicates whether the generated timestamp must be signed by CloudStreams using the configured keystore and signing alias. **Note:** For `signTimestamp` to work, you must ensure that a valid IS keystore and signing alias are configured in CloudStreams. For details, see "Setting the General Options" on page 26.	`False`
`useMilliSecondPrecision`	Boolean	Java Type: Boolean. Indicates whether the generated timestamp must have millisecond precision.	`True`
`MessageContext`	Object	Required. Java Type: `org.apache.axis2.context.MessageContext`. CloudStreams will place a `MessageContext` parameter into the pipeline before executing the IS service call.	`org.apache.axis2.context.MessageContext` instance

Parameter	Data Type	Description	Default Value
`actor`	String	Java Type: String. Indicates the value of the SOAP `actor` attribute if a new security header is being added to the SOAP request. If the request already has a security header with this attribute specified in it, then this value will not overwrite it.	`""`
`mustUnderstand`	Boolean	Java Type: Boolean. Specifies whether the security header will have the `mustUnderstand` attribute set to 0 or 1 (false / true). If the security header already has this attribute set, then this value will not overwrite it.	`False`

pub.cloudstreams.security.ws.AddWSAddressingHeaders

Adds WS-Addressing headers to a SOAP request sent by the client before CloudStreams forwards the request to the native service. This service includes the following input parameters:

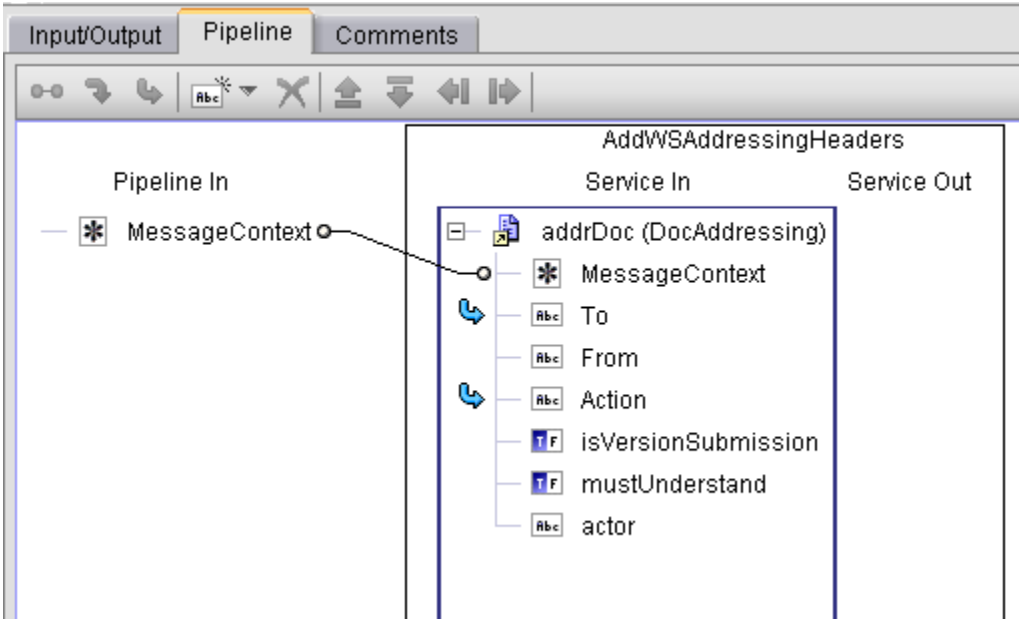
Parameter	Data Type	Description	Default Value
`isVersionSubmission`	Boolean	Java Type: Boolean. The WS-Addressing version that should be used. ■ If true, the WS-Addressing submission namespace `http://schemas.xmlsoap.org/ws/2004/08/addressing` will be used. ■ If false, the Final specification namespace `http://www.w3.org/2005/08/addressing` will be used.	`False`
`To`	String	Java Type: String. This value corresponds to the `/wsa:To` addressing header. You must specify a value that corresponds to the destination of the request message. If this value is not	

Parameter	Data Type	Description	Default Value
		specified, then depending on the `isVersionSubmission` parameter value, one of the following anonymous EPR values will be sent: If `isVersionSubmission` is set to true, the anonymous EPR value is: `http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous`. If `isVersionSubmission` is set to false, the anonymous EPR value is: `http://www.w3.org/2005/08/addressing/anonymous`.	
`From`	String	Java Type: String. This value corresponds to the `/wsa:From` addressing header and refers to the source of the message.	`""`
`Action`	String	Java Type: String. This value corresponds to the `/wsa:Action` addressing header. By default, this property has the same value as the operation on the virtual service being invoked (which will usually correspond to the same operation on the native service). But you can specify a different value corresponding to the native service being called.	URI identifying input operation corresponding to a WSDL port type being called on the virtual service.
`MessageContext`	Object	Required. Java Type: `org.apache.axis2.context.MessageContext`. CloudStreams will place a `MessageContext` parameter into the pipeline before executing the IS service call.	`org.apache.axis2.context.MessageContext` instance
`actor`	String	Java Type: String. Indicates the value of the SOAP `actor` attribute if a new security header is being added to the SOAP request. If the request already has a security header with this attribute	`""`

Parameter	Data Type	Description	Default Value
		specified in it, then this value will not overwrite it.	
`mustUnderstand`	Boolean	Java Type: Boolean. Specifies whether the security header will have the `mustUnderstand` attribute set to 0 or 1 (false / true). If the security header already has this attribute set, then this value will not overwrite it.	`False`

Example of using AddWSAddressingHeaders:

The sample service shown below is configured by providing the `MessageContext` parameter.



Using Context Variables in IS Services

CloudStreams provides predefined context variables, and you can declare your own custom context variables. You can use both predefined and custom context variables when you configure various processing steps of a virtual service. Specifically, you can use them in:

- An IS service that you create and invoke in the “In Sequence” and “Out Sequence” steps.
- In a routing rule that you create in a “Routing Rule” step of the type “Context-Based”.

This section discusses:

- “The Predefined Context Variables” on page 97.
- Custom context variables (see “The API for Context Variables” on page 100).

The Predefined Context Variables

You can use the predefined context variables listed below. Any context variable state defined during the inbound request processing steps will still be available during the outbound response processing steps.

Note:
To set, get or remove the predefined context variables, use the API for context variables provided by CloudStreams (see "The API for Context Variables" on page 100).

Note:
You do not need to declare the predefined context variables. If you attempt to declare an existing predefined context variable, an error will occur.

Context Variable Display Name	Context Variable Name	Description
Average Response	`AVG_SUCCESS_TIME`	The average amount of time it took the service to complete all invocations in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller.
Client IP Address	`INBOUND_IP`	The IP address used to send the request to CloudStreams.
CloudStreams Host Name	`HOSTNAME`	CloudStreams host name.
CloudStreams IP Address	`SERVER_IP`	CloudStreams IP address.
CloudStreams Target Name	`TARGET_NAME`	CloudStreams target name.
Consumer	`CONSUMER_APPLICATION`	The name of the consumer application accessing the service, if known.
Fault Count	`INTERVAL_FAULT_COUNT`	The number of invocations of the service resulting in a fault in the current interval.
Inbound Content Type	`MESSAGE_TYPE`	A Content-Type defined in `axis2.xml` for a message formatter. This value must be a key in the axis2 message formatters list, since it is used to control message serialization. (The valid choices are defined as attributes of `<messageFormatters/>` group in Integration Server's `axis2.xml` configuration.)

Context Variable Display Name	Context Variable Name	Description
Inbound HTTP Method	`INBOUND_HTTP_METHOD`	The HTTP method used by the client to send the request (**GET, POST, PUT, DELETE, CUSTOM**).
Inbound Protocol	`INBOUND_PROTOCOL`	The protocol (HTTP or HTTPS) of the request.
Maximum Response	`SLOWEST_SUCCESS_INVOKE`	The maximum amount of time it should take for the service to complete an invocation. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller.
Minimum Response	`FASTEST_SUCCESS_INVOKE`	The minimum amount of time it should take for the service to complete an invocation.
Outbound HTTP Method	`ROUTING_METHOD`	The HTTP method to be sent to the native service if the inbound HTTP method is custom. Otherwise, this value will be null. For more information, see "Changing the HTTP Method of a REST Virtual Service" on page 154.
Success Count	`INTERVAL_SUCCESS_COUNT`	The number of successful invocations of the service in the current interval.
Total Count	`INTERVAL_TOTAL_COUNT`	The total number of invocations (successful or unsuccessful) of the service in the current interval.
Virtual Service Name	`SERVICE_NAME`	Virtual service name.
(not displayed)	`BUILDER_TYPE`	A Content Type defined in `axis2.xml` for a message builder. This value must be a key in the axis2 message builders list, since it is used to control building of native service response messages. (The valid choices are defined as attributes of `<messageBuilders/>` group in Integration Server's `axis2.xml` configuration.)
(not displayed)	`INBOUND_REQUEST_URI`	A partial reference to a virtual service (for HTTP/HTTPS only). The protocol, host and port are not part of the value. For example, if the following virtual service is invoked: `http://user:5555/ws/TC1` then the expected value of this variable would be `/ws/TC1`.

Context Variable Display Name	Context Variable Name	Description

		For a REST or XML service, the URL might also include query string parameters. For example, if the following virtual service is invoked:
		
		`http://user:5555/ws/cars?vin=1234`
		
		then the expected value of this variable would be `/ws/cars?vin1234`.
		
		This is useful to know because by the time you are able to access the request inside of CloudStreams, the REST request would contain a top-level element that looks like this:
		
		`<vin>1234</vin>`
		
		So it is not obvious from an XSLT expression or an IS service callout what part of a REST request came in as a query parameter. Therefore, using this variable along with `INBOUND_HTTP_METHOD` and `INBOUND_PROTOCOL`, you can determine the exact entry point URI that was used when a CloudStreams virtual service was invoked.

(not displayed)	`NATIVE_PROVIDER_ERROR`	The "reason" returned by the native service provider in the case where it produced a SOAP fault. This variable will not contain CloudStreams-hosted errors such as policy violation errors; it will only contain the "reason" text wrapped in a SOAP fault.
(not displayed)	`OPERATION`	The service operation selected to execute a request.
(not displayed)	`PROTOCOL_HEADERS`	Contains a map of key-value pairs in the request, where the values are typed as strings. To get/set this variable, use pub.cloudstreams.ctxvar:getContextVariable (see "The API for Context Variables" on page 100).
(not displayed)	`SOAP_HEADERS`	(For use in IS services only.) Contains an array of the SOAP header elements in the request. To get/set this variable, use pub.cloudstreams.ctxvar:getContextVariable (see "The API for Context Variables" on page 100).

Context Variable Display Name	Context Variable Name	Description
(not displayed)	USER	The value defined for the Integration Server session executing the request message. If the request is not authenticated, it will use a default unprivileged account. Otherwise, it will set the Integration Server session to the user credentials used for transport security. In the case where a consumer is sending a request with both transport credentials (HTTP Basic authentication) and message credentials (WSS Username or X.509 token), the message credentials take precedence over the transport credentials.

The API for Context Variables

CloudStreams provides an API that you can use to:

- Set, get, declare and remove custom context variables.
- Set and get the predefined context variables. (It is not necessary (or even legal) to declare or remove the predefined context variables.)

CloudStreams provides the following Java services (described below):

- pub.cloudstreams.ctxvar:getContextVariable
- pub.cloudstreams.ctxvar:setContextVariable
- pub.cloudstreams.ctxvar:declareContextVariable
- pub.cloudstreams.ctxvar:removeContextVariable

The following sample flow services are described in this section as well:

- Sample Flow Service: Getting a Context Variable Value
- Sample Flow Service: Setting a Context Variable Value

pub.cloudstreams.ctxvar:getContextVariable

Use this Java service to retrieve a context variable's value and assign it to a pipeline variable. All parameter names are case-sensitive.

Parameter	Pipeline Type	Data Type	Description
MessageContext	in	Object ref	This object is inserted into the pipeline by CloudStreams.

Parameter	Pipeline Type	Data Type	Description
varName	in	String	Context variable name (predefined or custom). Examples: PROTOCOL_HEADERS, SOAP_HEADERS, mx:CUSTOM_VAR
serValue	out	Object ref	java.io.Serializable value. (Usually a string).

Notes on Getting and Setting the PROTOCOL_HEADERS and SOAP_HEADERS Variables

All context variable values are typed as either "string" or "int" except for the predefined context variables PROTOCOL_HEADERS and SOAP_HEADERS, which are of the type "IData". You can set/get values for PROTOCOL_HEADERS and SOAP_HEADERS in one of two ways:

1. **Set/get the entire structure.**

   To set the entire structure, you must:

   - Set the varName parameter in pub.cloudstreams.ctxvar:setContextVariable to PROTOCOL_HEADERS or SOAP_HEADERS.
   - Use the method RuntimeFacade.setContextVariableValue().

   To get the entire structure, you must:

   - Set the varName parameter in pub.cloudstreams.ctxvar:getContextVariable to PROTOCOL_HEADERS or SOAP_HEADERS.
   - Use the method RuntimeFacade.getContextVariableValue().

   If varName is set to PROTOCOL_HEADERS, you will get/set the entire IData structure containing all of the transport headers. The key is the transport header name (for example, Content-Type) and the value is a String. The IData object for PROTOCOL_HEADERS will contain a set of string values where each IData string key matches the header name in the transport headers map. The set of possible keys includes the HTTP v1.1 set of headers as well as any custom key-value pairs you might have defined.

   If varName is set to SOAP_HEADERS, you will get/set the entire IData structure containing all of the SOAP headers in the SOAP envelope. The key is the array position starting with "0", and the value is an Axiom OMElement containing that SOAP header block.

   Alternatively, you can set the varName parameter to address a specific element in the array. For example, setting it to PROTOCOL_HEADERS[Content-Type] would apply to the Content-Type transport header. Similarly, setting it to SOAP_HEADERS[0] would return a String representation of the first SOAP header block (as opposed to an Axiom OMElement).

2. **Set/get a nested value.**

   Set a nested value in one of the following ways:

   - Set the varName parameter in pub.cloudstreams.ctxvar:setContextVariable to PROTOCOL_HEADERS[arrayElement], where arrayElement refers to a specific element. For

example, PROTOCOL_HEADERS[Content-Type] or SOAP_HEADERS[0] (to indicate the first array element in the set).

- Alternatively, use the method RuntimeFacade.setContextVariableValue(). You would use this method only if you are writing a Java service and you want to access it through the Java source code.

Get a nested value in one of the following ways:

- Set the varName parameter in pub.cloudstreams.ctxvar:getContextVariable to PROTOCOL_HEADERS[arrayElement], where *arrayElement* refers to a specific element. For example, PROTOCOL_HEADERS[Content-Type] or SOAP_HEADERS[0] (to indicate the first array element in the set).
- Alternatively, use the method RuntimeFacade.getContextVariableValue(). You would use this method only if you are writing a Java service and you want to access it through the Java source code.

You can set/get a nested value inside PROTOCOL_HEADERS and SOAP_HEADERS via an additional keyName. In this case, the object reference will not be an IData object.

- For PROTOCOL_HEADERS, the keyName must match the transport header name in a case-sensitive manner (for example, PROTOCOL_HEADERS[Content-Type] or PROTOCOL_HEADERS[Authorization]). In this case, the Serializable value will be a string.
- For SOAP_HEADERS, the keyName must match the 0-based array element. If a request has a SOAP security header element (</wsse:Security>), then it would be addressed as SOAP_HEADERS[0]. In this case, the element will be in its string format.

pub.cloudstreams.ctxvar:setContextVariable

Use this Java service to set a value on a context variable. The pipeline variable containing the context variable value should be an object reference that implements java.io.Serializable. All parameter names are case-sensitive.

Parameter	Pipeline Type	Data Type	Description
MessageContext	in	Object ref	This object is inserted into the pipeline by CloudStreams.
varName	in	String	Context variable name (predefined or custom). Examples: PROTOCOL_HEADERS, SOAP_HEADERS, mx:CUSTOM_VAR
serValue	in	Object ref	Java.io.serializable value. (Usually a string.)

pub.cloudstreams.ctxvar:declareContextVariable

Use this Java service to declare a *custom* context variable. All custom-defined context variables must be declared in a custom namespace that is identified by using the prefix `mx` (for example, `mx:CUSTOM_VARIABLE`). All parameter names are case-sensitive.

Note:
It is not legal to use this service to declare the predefined context variables; you can only declare custom variables.

Parameter	Pipeline Type	Data Type	Description
`ctxVr`	in	Object ref	The document type defining the context variable object. Use the ctxVar Document Type provided in the Java service pub.cloudstreams.ctxvar:ctxVar and map it to this input variable. Define the name of the context variable (for example, mx:CUSTOM_VARIABLE), the schema_type (string or int), and isReadOnly (true or false).
`ctxVar`	out	Object ref	The set Context variable document type.
`varNameQ`	out	Object ref	javax.xml.namespace.QName value. The QName of the variable.

Note the following:

- After declaring the context variable, you can use the setContext variable to set a value on the context variable.
- You do *not* need to declare the following kinds of context variables:
  - The predefined variables provided by CloudStreams. If you attempt to declare an existing predefined context variable, an error will occur.
  - Any custom context variable that you define in a routing rule that you create in a "Context-Based" Routing Rule step.
- Any custom context variables that you explicitly declare in source code using the API will have a declaration scope of SESSION.
- Any custom context variable's state that is defined during the inbound request processing steps will still be available during the outbound response processing steps.
- All context variable values are typed as either "string" or "int" (excluding the `SOAP_HEADERS` and `PROTOCOL_HEADERS` variables, which are of the type "IData").
- Valid names should be upper case (by convention) and must be a valid Java Identifier. In general, use alpha-numerics, `$` or _ symbols to construct these context names. Names with punctuation, whitespace or other characters will be considered invalid and will fail deployment.

- All custom context variables must be declared in a custom namespace that is identified by using an `mx` prefix (for example, `mx:CUSTOM_VARIABLE`).
- To reference a custom context variable in a flat string, you need to prepend a `$` symbol to the context variable name to indicate that variable's value should be referenced. Think of this usage as being similar to the "&" address operation for C variables.

  An expression that references a custom context variable might look like this:

  `$mx:TAXID=1234` or `$mx:ORDER_SYSTEM_NAME="Pluto"`

  Notice that the values of the data type `int` are not enclosed in quotation marks, while the values of the data type `string` are. The quotation marks are only needed if a context variable expression (as opposed to a reference) is defined.
- Referencing an undefined context variable does not result in an error.
- Once a variable has been declared it cannot be declared again.

pub.cloudstreams.ctxvar:removeContextVariable

Use this Java service to remove a *custom* context variable from a request/response session. All parameter names are case-sensitive.

Note:
It is not legal to use this service to remove any predefined context variables; you can only remove custom variables.

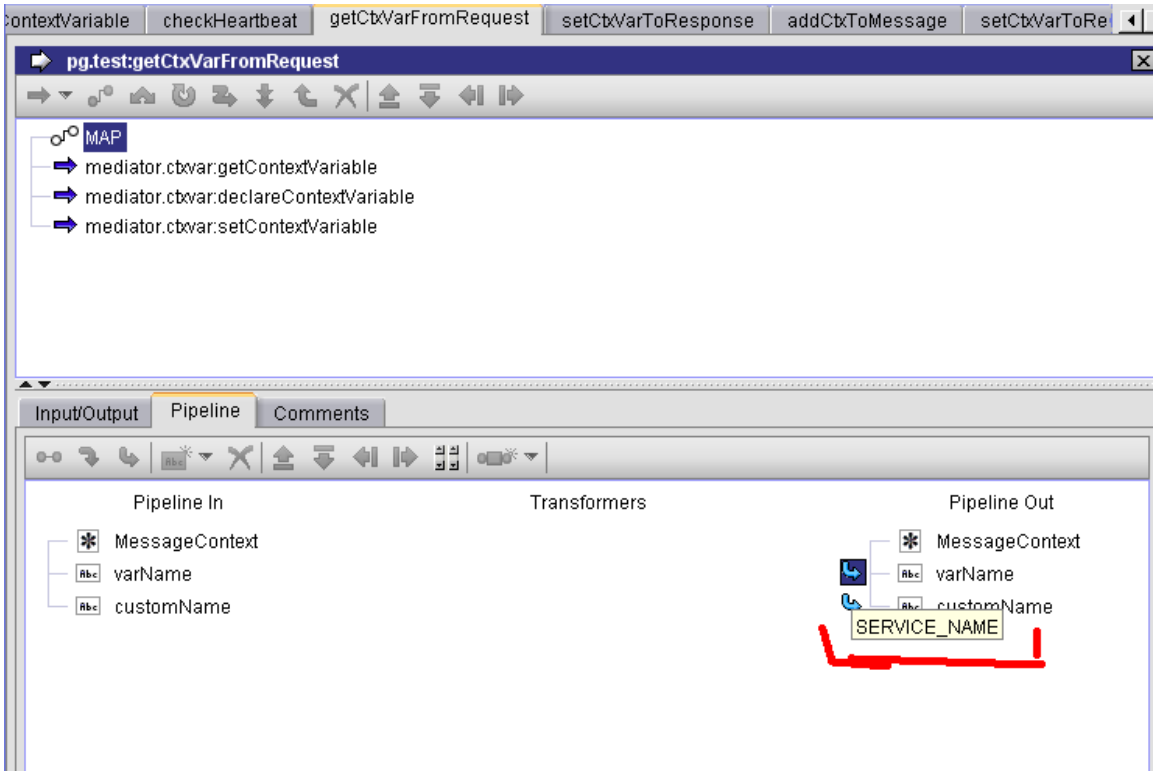
Note:
Attempting to remove a non-existent context variable will not result in an error.

Parameter	Pipeline Type	Data Type	Description
`MessageContext`	in	Object ref	This object is inserted into the pipeline by CloudStreams.
`varName`	in	String	Custom context variable name. Example: `mx:CUSTOM_VAR`.

Sample Flow Service: Getting a Context Variable Value

This flow service gets the value of a custom context variable from a request. The service declares a context variable using the name defined in the pipeline variable `customName` (that is, `mx:COMP_TEST`). It is hard-coded to store the predefined context variable `SERVICE_NAME` in this custom context variable name.

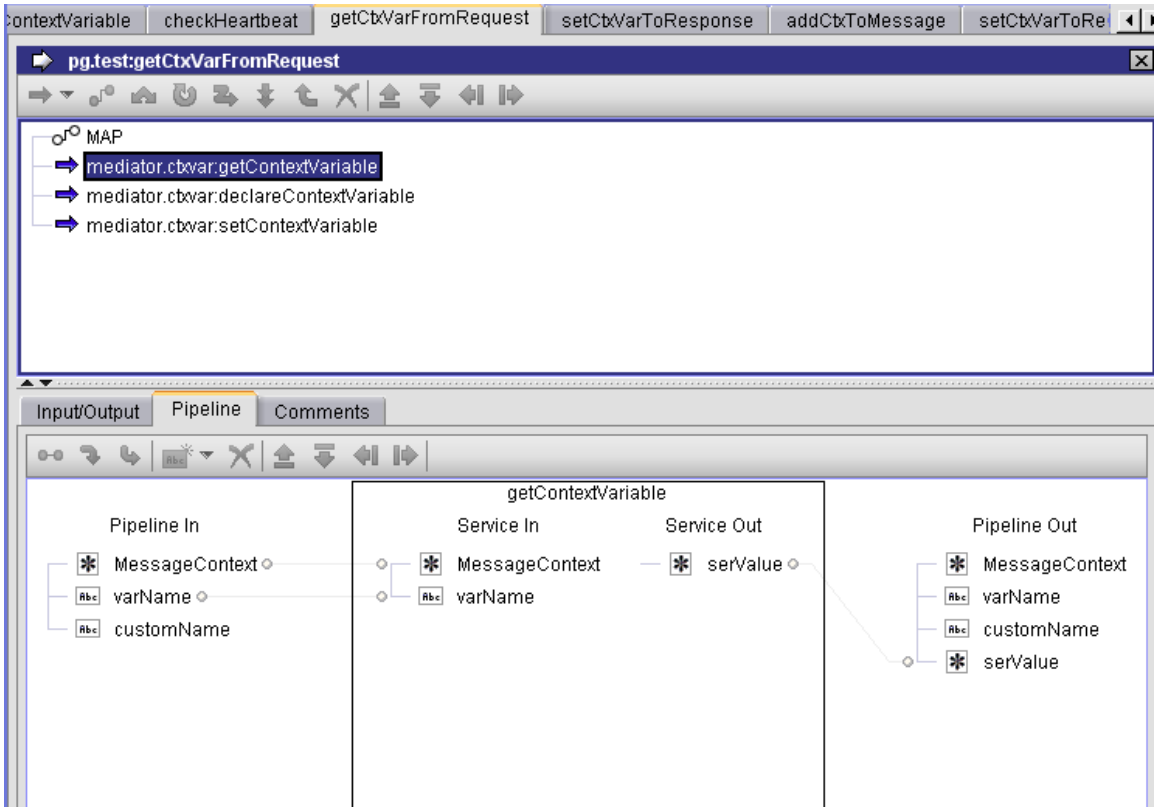
Step 1: Setting varName and customName



In this flow service, the following pipeline variables that are hard-coded as follows:

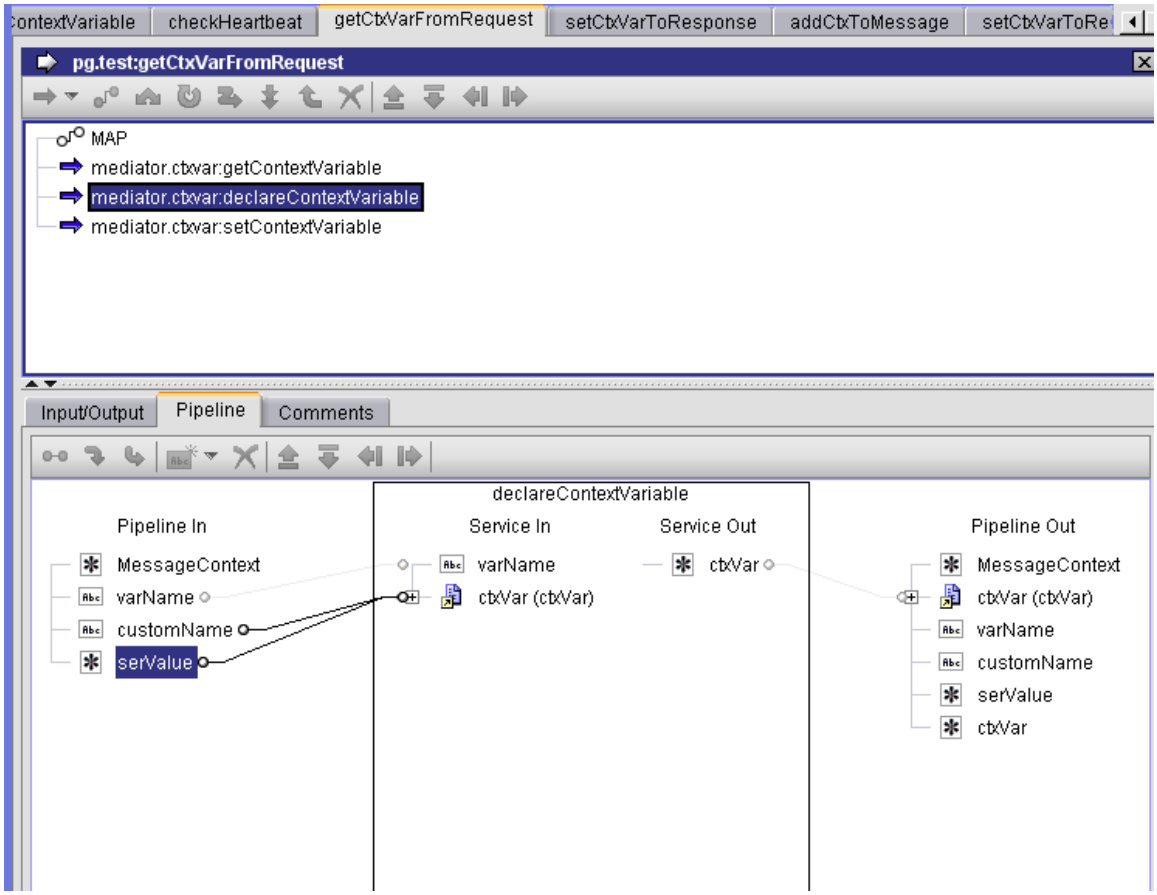
- `varName` is set to the predefined context variable `SERVICE_NAME`.
- `customName` is set to `mx:COMP_TEST`, and `SERVICE_NAME` is stored in `customName` in the "Pipeline Out".

Step 2: Calling the pub.cloudstreams.ctxvar:getContextVariable service



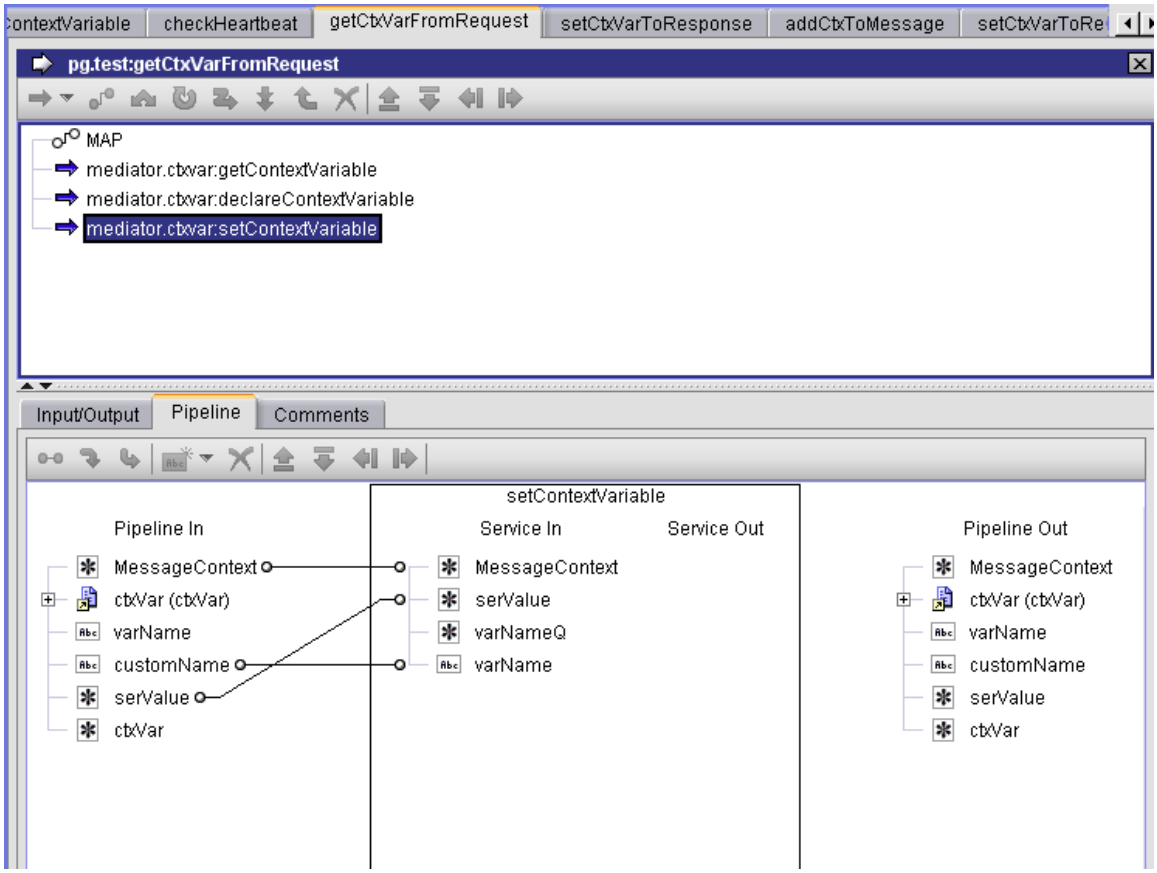
The call to `pub.cloudstreams.ctxvar:getContextVariable` looks up the context variable value for the "Pipeline In" variable `varName` and sets the `Serializable` value in the "Pipeline Out" variable `serValue`.

Step 3: Calling the pub.cloudstreams.ctxvar:declareContextVariable service



The call to `pub.cloudstreams.ctxvar:declareContextVariable` takes the serialized value we looked up for `varName` and assigns it to a newly declared custom context variable named `ctxVar`.

Step 4: Calling the pub.cloudstreams.ctxvar:setContextVariable service



The call to `pub.cloudstreams.ctxvar:setContextVariable` sets the context variable value back into `MessageContext` so it will be available for the rest of the service invocation steps.

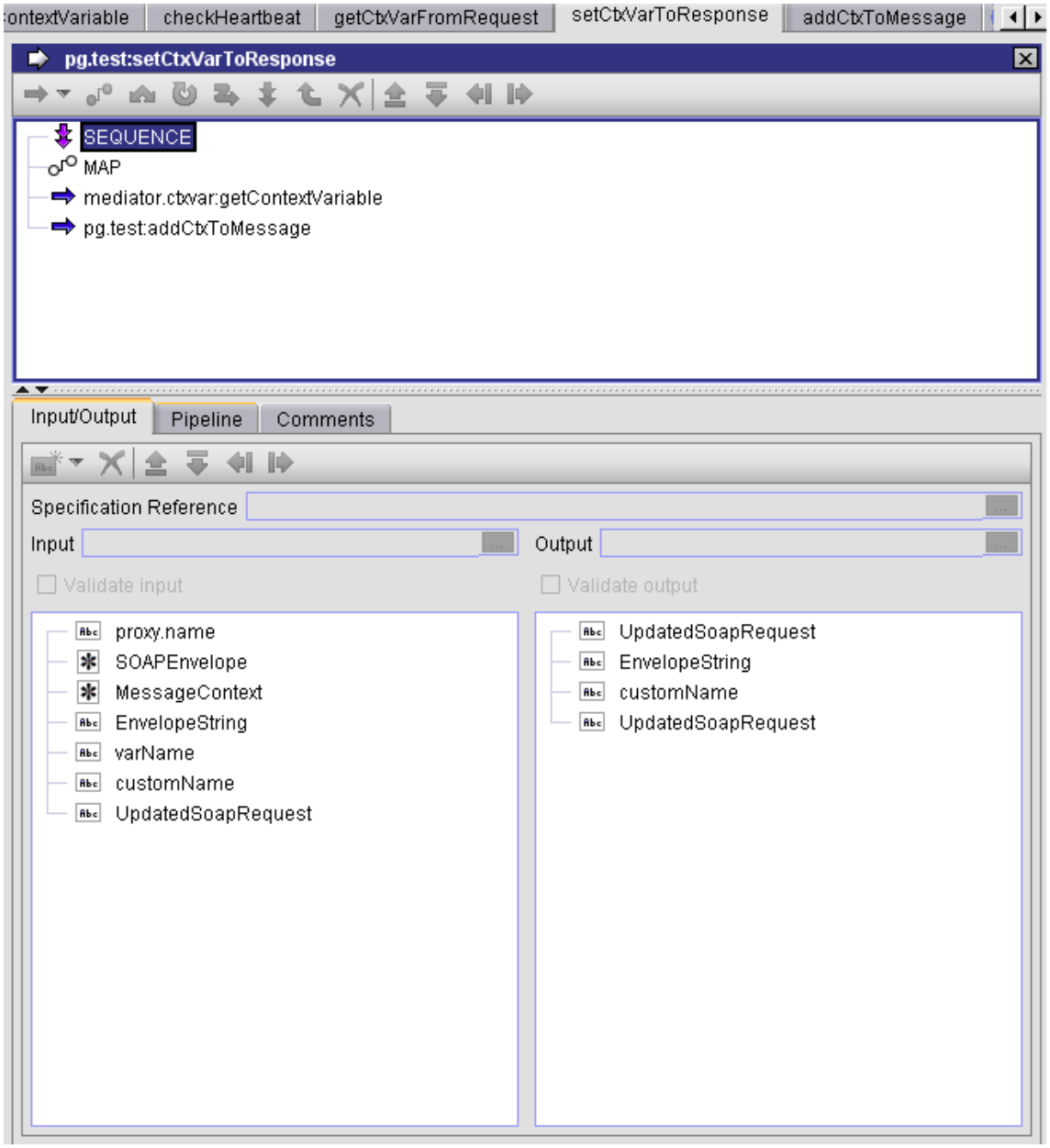
Sample Flow Service: Setting a Context Variable Value

This flow service sets the value of a custom context variable to be used in a response.

This flow service declares a pipeline variable named `customName`, which is set to the value `mx:COMP_TEST`.

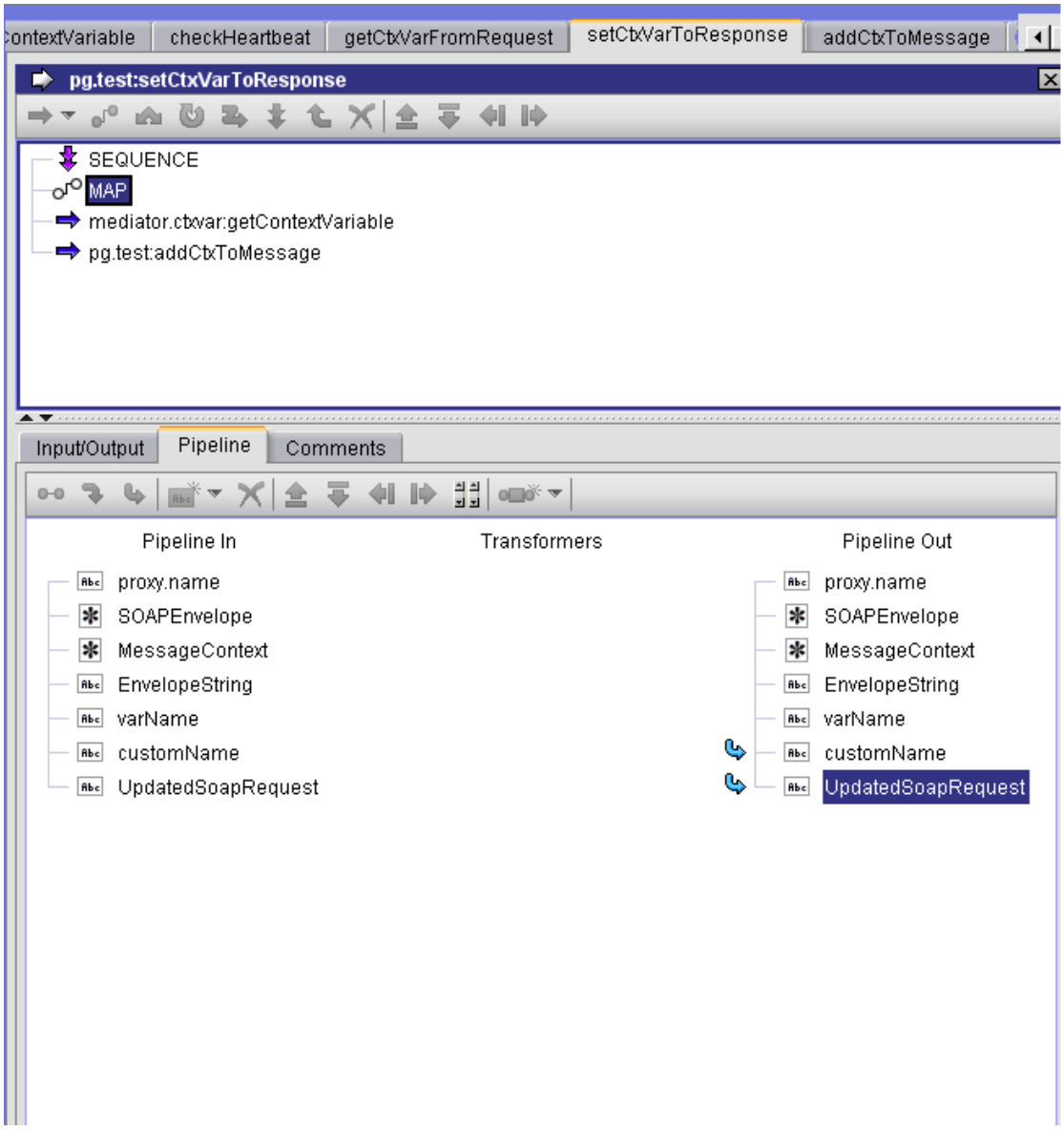
This flow service will retrieve the context variable for `customName` and create an element for its context variable value in the response message return to the consumer.

Step 1: Declaring customName



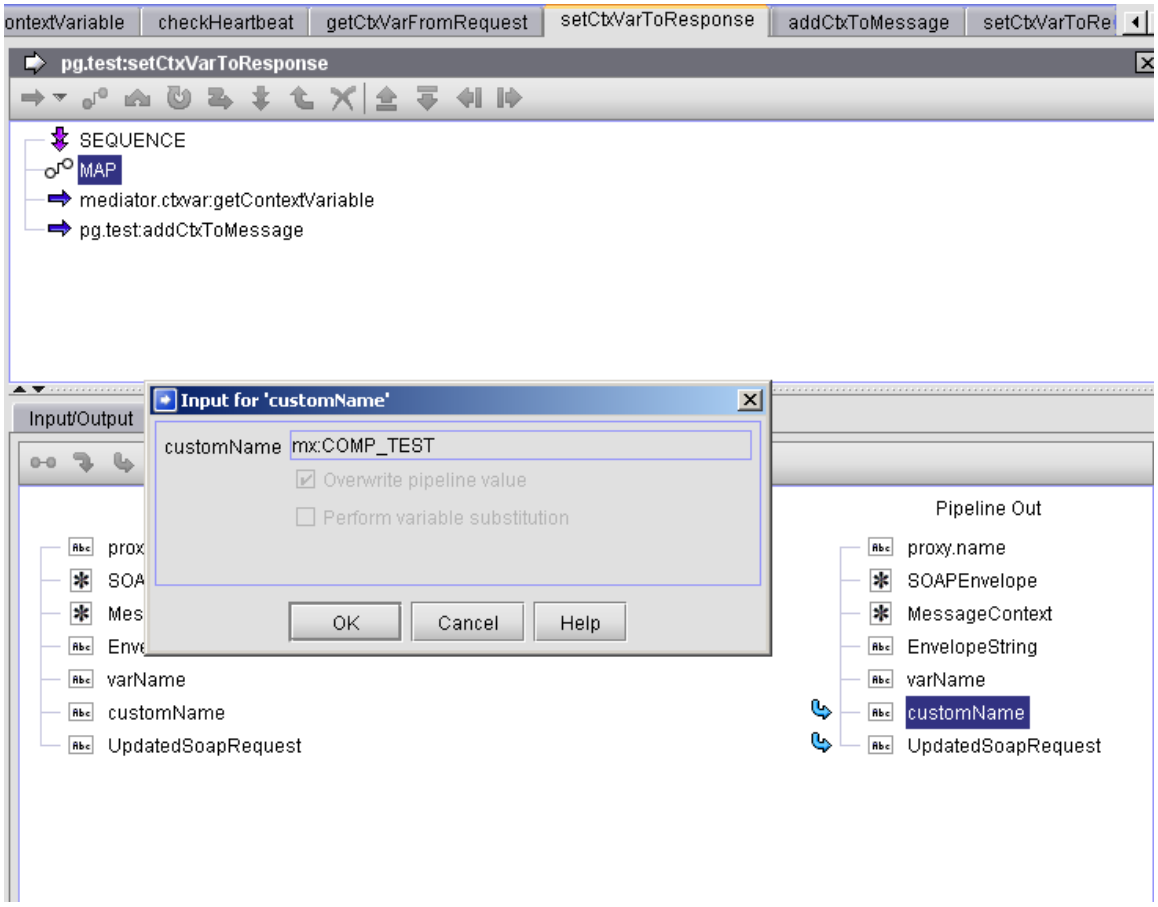
We define the `customName` variable to be `mx:COMP_TEST` so we can use this variable to lookup the custom variable name that was seeded in the previous example.

Step 2: Setting customName to mx:COMP_TEST



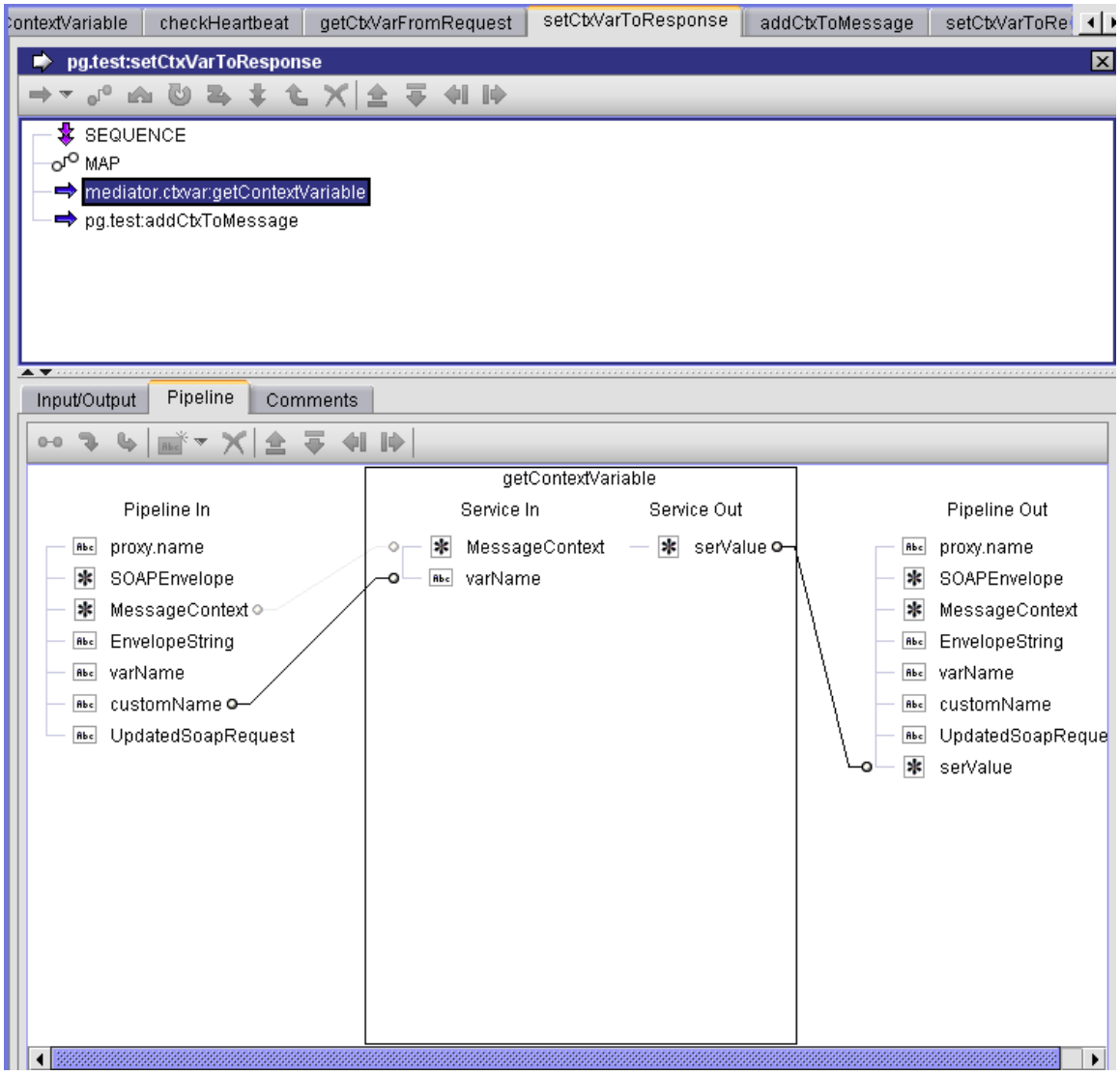
Clicking on the `customName` pipeline variable will display the name.

Step 3: Displaying the value of customName



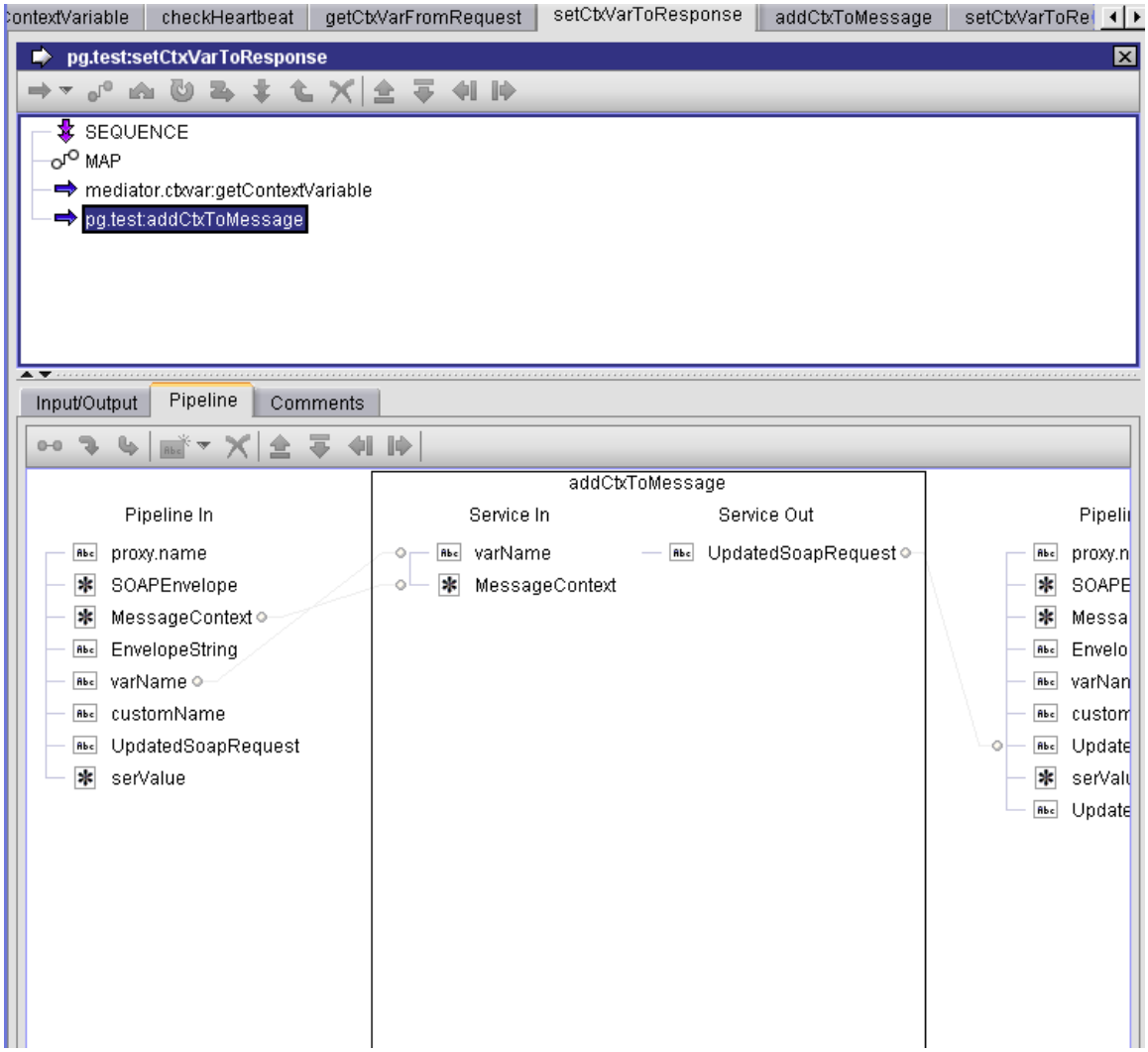
The call to `pub.cloudstreams.ctxvar:getContextVariable` retrieves the value of the custom context variable from the context variable map.

Step 4: Calling pub.cloudstreams.ctxvar:getContextVariable



This is just a sample Java service that takes the context variable and creates a top-level element in the response message using the same name and value.

Step 5: Sample service using the context variable



Creating a New Connector Virtual Service (SOAP)

A connector virtual service will:

- Handle the provider's response.
- Log the requests/responses.

CloudStreams provides a default connector virtual service for each metadata handler:

- The service `WmCloudStreams.SoapVS` (for the SOAP handler).
- The service `WmCloudStreams.RestVS` (for the REST handler).

The default services are located in the sample CloudStreams Governance project in the WmCloudStreams package. You cannot modify them.

Each default connector virtual service has a default policy (named `Logging Policy`), which logs all requests/responses and their payloads to a database. You cannot modify the default policy. Alternatively, you can create additional connector virtual services with custom policies.

If you create a connector virtual service with a custom policy, you can only include the actions in the "Monitoring" or "Additional" action categories; you cannot include the "WS-SecurityPolicy 1.2" actions. For example, you might want to create a custom policy that monitors run-time performance, customizes how the service invocations are logged, validates response messages against an XML schema, or optimizes server traffic. For complete details, see the chapter "Policies" on page 159.

To create a new connector virtual service (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, right-click the CloudStreams Governance project and click **New > Connector Virtual Service** (or expand the project, right-click the **Connector Virtual Services** folder and click **New Connector Virtual Service**).

3. Complete the following fields in the New Connector Virtual Service wizard and click **Finish**.

Option	Description
Project	Click **Browse** and select a CloudStreams Governance project in which to create the connector virtual service.
Name	Assign a name for the service. Unlike native services, the names of virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the virtual service name, then the names of the services themselves must not contain characters that are invalid in virtual service names. **Note:** If you want to change the service name after it has been created, right-click the service name in the **Virtual Services** folder and select **Rename**.
Version	The version is always set to 1.0.
Type	Select **SOAP**.
Description	Optional. A description for the service. This description appears when a user displays a list of services in the user interface.

The new connector virtual service is added to the CloudStreams Governance project, in the **Connector Virtual Services** folder.

The Properties of a Connector Virtual Service (SOAP)

To view the properties of a connector virtual service (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name. (If the Properties view is not already open, click **Window > Show View > Other > General > Properties**.)

3. The **General** page in the Properties view displays the following properties:

Option	Description
Name	(Read-only field.) The service name. You can change the service name at any time by right-clicking the name in the **Connector Virtual Services** folder and clicking **Rename**.
Service Type	(Read-only field.) **SOAP**.
Created/Last Modified	(Read-only field.) The service's creation/modification timestamps.
Target Namespace	(Read-only field.) The value derived from the `targetNamespace` attributes of the WSDL's definition element.
Namespaces	Click the button next to this field to view the service's available namespaces (such as `wsdl`, `xsd`, `soap`, etc.).
Version	The version is always set to 1.0.
WSDL URL	(Read-only field.) The URL of the service's WSDL.
Description	You can change the service description.

4. View additional properties in the **Advanced** page.

Option	Description
Name	(Read-only field.) The service name.
Type	(Read-only field.) The type of the service (SOAP).
Target Namespace	(Read-only field.) The value derived from the `targetNamespace` attributes of the WSDL's definition element.
WSDL URL	Click this URL to display the contents of the service's abstract WSDL. This is an internal WSDL used by CloudStreams primarily to

Option	Description
	manage deployment of connector virtual services to a CloudStreams target. The native provider's WSDL that was defined for the connector is what is used when the connector service is configured.
Namespaces	Click the button next to this field to view the service's available namespaces (such as `wsdl`, `xsd`, `soap`, etc.).
Version	(Read-only field.)
Description	(Read-only field.) The service description.
VSD	Click the button next to this field to view the service's "virtual service definition" (VSD). This button is disabled until you deploy the connector virtual service to a CloudStreams server. When you deploy the connector virtual service to a CloudStreams server, CloudStreams generates an XML document called a *virtual service definition (VSD)*. The VSD defines the connector virtual service for CloudStreams, and contains all the resources required to deploy the connector virtual service to a CloudStreams server, including the policy that applies to the service. You cannot edit the VSD. **Note:** If multiple policies apply to the service, CloudStreams combines all those policies into a single policy known as the *effective policy*. The effective policy is a simple UNION of the run-time actions specified in all policies that apply to a service. To create the effective policy, CloudStreams evaluates the combined list of actions from all policies, using a set of internal rules known as Policy Conflict Resolution rules. For details, see "Policy Conflict Resolution Rules" on page 195.
Applicable Policies	Click the button next to this field to view a list of the active policies that apply to this service. Any inactive policy that applies to the service is not listed. To change the list of applicable policies, see "Modifying Policies" on page 164.
Endpoint	(Read-only field). The endpoint is resolved when the connector service is configured and the user selects an enabled managed connection pool.
Deployed Status	(Read-only field.) Indicates whether the service is **Deployed**, **Undeployed** or **Not Deployed** (which is the initial status before you deploy the service). To deploy or undeploy a service, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Resource	Not applicable to SOAP services.

Managing a Connector Virtual Service (SOAP)

You can rename, delete, deploy or undeploy a connector virtual service, and change the list of policies that apply to the it.

To manage a connector virtual service (SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project.

3. Right-click the connector virtual service name and choose any of the following tasks from the context menu:

Option	Description
Rename	Use this option to assign a new name to the service. The service must be undeployed before you can rename it (check the **Deployed Status** field in the Advanced page of the service's Properties view). Unlike native services, the names of connector virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the connector virtual service name, then the names of the services themselves must not contain characters that are invalid in connector virtual service names.
Delete	Deletes the connector virtual service from CloudStreams.
Deploy	Deploys the connector virtual service to a CloudStreams server target. For more information, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Undeploy	Undeploys the connector virtual service from a CloudStreams server target.

4. To change the list of policies that apply to the connector virtual service, see "Modifying Policies" on page 164.

The "In Sequence" Step (Connector Virtual Service, SOAP)

The "In Sequence" step can contain the following sub-steps:

- The "Entry Step" (required), which specifies the protocol of the requests that the connector virtual service will accept.

- The “Routing Rule” step (required), which specifies how the connector virtual service will route the service requests to the native service endpoint.
- The “Invoke IS Service” step (optional), which pre-processes the request message before it is submitted to the native service.

The “Entry Step” (Connector Virtual Service, SOAP)

This step specifies the protocol of the requests that the connector virtual service will accept. You cannot modify this step, except for it name.

To edit the “Entry Step” (connector virtual service, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the connector virtual service name.

   The virtual service editor displays the three main processing steps: **In Sequence**, **Out Sequence** and **Error Sequence**.
3. Expand **In Sequence**.

   By default, the **In Sequence** step contains the sub-steps **Entry Step** and **Routing Rule**. (You can add the optional **Invoke IS Service** step, as described later.)
4. Click **Entry Step** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Entry Step**.
Protocol	(Read-only field.) The protocol of the requests that the connector virtual service will accept. The value **Local** means that the service can only be called from Cloud Connector Services.

The “Routing Rule” Step (Connector Virtual Service, SOAP)

This step specifies how the connector virtual service will route the requests to the native service endpoint. You cannot modify this step, except for its name.

To edit the Routing Rule step (connector virtual service, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the connector virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	Read-only field.) **Protocol**. You cannot change the protocol over which the connector virtual service will accept requests.
Routing Rule	(Read-only field.) The value **Connection Pool** means that the requests are sent to the provider using the CloudStreams connection pool. For more information about connection pools, see the documentation specific to your CloudStreams cloud provider (for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*).

The "Invoke IS Service" Step (Inbound, Connector Virtual Service, SOAP)

This optional step invokes a user-defined Integration Server flow service. An IS service is a user-defined Integration Server flow service that you can invoke in:

- The "In Sequence" step, to pre-process the request messages before CloudStreams submits the requests to the native service.
- The "Out Sequence" step, to pre-process the response messages from the native service before CloudStreams returns the responses to the consuming application.

You create an "Invoke IS Service" step for a connector virtual service the same way you create one for a virtual service. For details, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69, or "The Invoke IS Service Step (Outbound, SOAP)" on page 83.

The "Out Sequence" Step (Connector Virtual Service, SOAP)

You configure the service's "Out Sequence" step (optional) to manipulate the response messages from the native service before CloudStreams returns the responses to the consuming applications. This step can include the "Invoke IS Service" step (optional), which pre-processes the response

message from the native service provider before it is returned to the consuming application. See "The Invoke IS Service Step (Inbound, Connector Virtual Service, SOAP)" on page 119.

The "Error Sequence" Step (Connector Virtual Service, SOAP)

The default connector virtual services are configured to return the native service provider's service fault, if one is available. CloudStreams will send whatever content it received from the native service provider. You can optionally invoke IS services to pre-process or post-process the error messages.

To configure the Error Sequence step (connector virtual service, SOAP)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the connector virtual service name.

3. In the virtual service editor, expand **Error Sequence**.

4. Click the **Error Messaging** sub-step and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Error Messaging**.
Processing Method	Optionally select either of the following: ■ **Pre-Processing**: Select this option if you want to invoke an IS service to manipulate the response message before the Error Sequence step is invoked. The IS service will have access to the response message context (the axis2 `MessageContext` instance) before it is updated with the custom error message. For example, you might want to send emails or perform custom alerts based on the response payload. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69. ■ **Post-Processing**: Select this option if you want to invoke an IS service to manipulate the service fault after the Error Sequence step is invoked. The IS service will have access to the entire service fault and the custom error message. You can make further changes to the fault message structure, if needed. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69.

Creating a New Virtual Service (REST)

To create a new virtual service (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, right-click the CloudStreams Governance project and click **New > Virtual Service** (or expand the project, right-click the **Virtual Services** folder and click **New Virtual Service**).

3. In the New Virtual Service wizard, complete the following fields.

Option	Description
Project	Click **Browse** and select a CloudStreams Governance project in which to create the service.
Name	Assign a name for the service. Unlike native services, the names of virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the virtual service name, then the names of the services themselves must not contain characters that are invalid in virtual service names. **Note:** If you want to change the service name after it has been created, right-click the service name in the **Virtual Services** folder and select **Rename**.
Version	The version is always set to 1.0.
Service Type	Select **REST**.
Description	Optional. A description for the service. This description appears when a user displays a list of services in the user interface.

4. Click **Finish**.

   The new virtual service is added to the CloudStreams Governance project, in the **Virtual Services** folder.

5. Click the new virtual service name in the CloudStreams Governance project.

   The Properties view is displayed at the bottom of the editor.

6. On the **Advanced** page in the Properties view, click the **Resource** button.

7. In the REST Resources dialog box that appears, add a REST resource that the service will access by completing the following fields.

Option	Description
Resource	Specify the name of the REST resource that the service will access. **Note:** To add a REST resource to the service after the service has been created, right-click the service name and click **Attach Resource**.
HTTP Methods	(Read-only field.) The HTTP methods that the virtual service should be allowed to perform on the REST resource (**GET**, **POST**, **PUT**, **DELETE**). You specify the HTTP methods when you configure the virtual service's Entry Step.
Content Type	(Read-only field.) **Application/xml** or **Application/json**.
Query String	Specify the search string as required. Any text typed in is not case sensitive. This field is used only for documentation purposes at this time. It is not included in the service's "virtual service definition" (VSD) that is deployed to CloudStreams, so this value has no impact at runtime.
Import from	Import the schemas that the virtual service will use. Choose one of the following options. (To select multiple schemas, click the plus button to add a row.) ■ **URL**: Specify the URL of a schema and click the **Add** button. Optionally choose the following option: ■ **URL authentication**: If you have specified a URL, and the site you want to access via the URL requires user authentication, select this option. This opens an Authentication sub-dialog in which you can enter a user name and password for authentication at the URL site. ■ **File**: Click **Browse** to select a schema from your local file system.

8. Click **OK**.

The Properties of a Virtual Service (REST)

To view or modify the properties of a virtual service (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name. (If the Properties view is not already open, click **Window > Show View > Other > General > Properties**.)

3. The **General** page in the Properties view displays the following properties:

Option	Description
Name	(Read-only field.) The service name. You can change the service name at any time by right-clicking the name in the **Virtual Services** folder and clicking **Rename**.
Service Type	(Read-only field.) **REST**.
Created/Last Modified	(Read-only field.) The service's creation/modification timestamps.
Target Namespace	(Read-only field.) The value is derived from the `targetNamespace` attributes of the WSDL's definition element.
Namespaces	Specify other namespaces (such as `wsdl`, `xsd`, etc.).
Version	The version is always set to 1.0.
WSDL URL	(Read-only field.) The URL of the REST resource's abstract WSDL.
Description	You can change the service description.

4. View additional properties on the **Advanced** page.

Option	Description
Name	(Read-only field.) The service name.
Type	(Read-only field.) **REST**.
Target Namespace	(Read-only field.) The value is derived from the `targetNamespace` attributes of the WSDL's definition element.
WSDL URL	Click this URL to display the contents of the REST resource's abstract WSDL. If a WSDL file was not added, this will be empty.
Namespaces	Specify other namespaces (such as `wsdl`, `xsd`, etc.).
Version	The version is always set to 1.0.
Description	(Read-only field.) The service description.
VSD	Click the button next to this field to view the service's “virtual service definition” (VSD). This button is disabled until you deploy the virtual service to a CloudStreams server.

Option	Description
	When you deploy the virtual service to a CloudStreams server, CloudStreams generates an XML document called a *virtual service definition (VSD)*. The VSD defines the virtual service for CloudStreams, and contains all the resources required to deploy the virtual service to a CloudStreams server, including the policy that applies to the service. You cannot edit the VSD. **Note:** If multiple policies apply to the service, CloudStreams combines all those policies into a single policy known as the *effective policy*. The effective policy is a simple UNION of the run-time actions specified in all policies that apply to a service. To create the effective policy, CloudStreams evaluates the combined list of actions from all policies, using a set of internal rules known as Policy Conflict Resolution rules. For details, see "Policy Conflict Resolution Rules" on page 195.
Applicable Policies	Click the button next to this field to view a list of the active policies that apply to this service. Any inactive policy that applies to the service is not listed.
Endpoint	Click the button next to this field to view the endpoint to which the virtual service is deployed, if applicable.
Deployed Status	(Read-only field.) Indicates whether the service is **Deployed**, **Undeployed** or **Not Deployed** (which is the initial status before you deploy the service). To deploy or undeploy a service, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Resource	Click the button next to this field to view the REST resource that the service will access. For details, see "Creating a New Virtual Service (REST)" on page 121. **Note:** To add a REST resource to the service, right-click the service name and click **Attach Resource**.

Managing a Virtual Service (REST)

You can rename, delete, deploy or undeploy a virtual service, and attach a REST resource to it.

To manage a virtual service (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project.

3. Right-click the virtual service name and choose one of the following tasks from the context menu:

Option	Description
Rename	Use this option to assign a new name to the service. The service must be undeployed before you can rename it (check the **Deployed Status** field in the Advanced page of the service's Properties view). Unlike native services, the names of virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the virtual service name, then the names of the services themselves must not contain characters that are invalid in virtual service names.
Delete	Deletes the virtual service from CloudStreams.
Attach Resource	Use this option to attach a REST resource to an existing virtual service. The new resource will override the old one.
Deploy	Deploys the virtual service to a CloudStreams server target. For more information, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Undeploy	Undeploys the virtual service from a CloudStreams server target.

4. To change the list of policies that apply to the virtual service, see "Modifying Policies" on page 164.

The "In Sequence" Step (REST)

You configure the service's "In Sequence" step to manipulate the request messages. This step can include the following sub-steps:

- The "Entry Step" (required), which specifies the protocol (HTTP or HTTPS) of the requests that the virtual service will accept, and the HTTP methods that the virtual service should be allowed to perform on the REST resource.
- The "Transform" step (optional), which performs an XSLT message transformation on the request message before it is submitted to the native service.
- The "Invoke IS Service" step (optional), which pre-processes the request message before it is submitted to the native service.
- The "Routing Rule" step (required), which specifies how the virtual service will route the requests to the native service endpoint(s). There are four ways to route HTTP or HTTPS requests:

- "Straight Through" routing (to route requests directly to the native service endpoint).
- "Context-Based" routing (to route specific types of messages to specific endpoints according to context-based routing rules).
- "Content-Based" routing (to route specific types of messages to specific endpoints based on specific values that appear in the request message).
- "Load Balancing" routing (to distribute requests among multiple endpoints).

The "Entry Step" (REST)

The Entry Step (required) specifies the protocol (HTTP or HTTPS) over which the virtual service will accept requests, and the HTTP methods (GET, POST, PUT, DELETE) that the virtual service should be allowed to perform on a REST resource.

This step allows you to bridge protocols between the consuming application and the native service. For example, suppose you have a native service that is exposed over HTTPS and a consuming application that submits SOAP requests over HTTP. In this situation, you can configure the virtual service's Entry Step to accept HTTP requests and configure its Routing Rule step to route the request to the Web service using HTTPS.

To configure the Entry Step (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

   The virtual service editor displays the three main processing steps: **In Sequence**, **Out Sequence** and **Error Sequence**.

3. In the virtual service editor, expand **In Sequence**.

   By default, the **In Sequence** step contains the sub-steps **Entry Step** and **Routing Rule**. (You can add the optional **Transform** and **Invoke IS Service** steps, as described later.)

4. Click **Entry Step** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Entry Step**.
HTTP Methods	The HTTP methods that the virtual service should be allowed to perform on the REST resource (GET, POST, PUT, DELETE).

Option	Description
	It is important to specify all the HTTP methods that are supported for the virtual service. For example, if the virtual service is deployed to CloudStreams and only the GET method was selected in the virtual service definition, then CloudStreamswill only permit GET invocations. A POST request will be rejected with a return of statusCode 405 even if the native service happens to support POSTs. It is also important for the native service to return the correct `Content-Type` header with its responses. At run time, CloudStreams determines the type of a request message based on the message's HTTP method and its Content-Type. See "HTTP Method/Content-Type Combinations" on page 153.
Protocol	The protocol (HTTP or HTTPS) over which the virtual service will accept requests. To specify HTTPS, select both **HTTP** and **SSL**. CloudStreams supports HTTP v1.1 only.

The "Transform" Step (Inbound, REST)

The optional "Transform" step specifies how the request message is to be transformed before it is submitted to the native service.

As long as a consumer sends a REST request with the proper Content-Type (see "HTTP Method/Content-Type Combinations" on page 153) and as long as the HTTP method and endpoint URI are in the expected form, then CloudStreams can determine the correct service and operation; in this case, no message transformation is required. However, in some cases a virtual REST service might need to transform XML messages.

For example, you might need to accommodate differences between the message content that a consuming application is capable of submitting and the message content that a native service expects. For example, if the consuming application submits an order record using a slightly different structure than the structure expected by the native service, you can use the Transform step to transform the record submitted by the consuming application to the structure required by the Web service.

In this case, you would need to create two Transform steps:

- One in the "In Sequence" step, to transform the request messages into the format required by the native service, before CloudStreams sends the requests to the native services. To do this, you pass the message to an XSLT transformation file. (Additionally in this case, the transformation is required if the virtual service has a schema validation policy that validates the requests.)
- One in the "Out Sequence" step, to transform the native service's response messages into the format required by the consumer applications, before CloudStreams returns the responses to the consumer applications.

To add the "Transform" step (inbound, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Right-click **In Sequence** and select **Transform**.

   The Transform step is added under the Entry Step. You cannot change the order of the steps.

4. Click **Transform** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only.) **Transform**.
XSLT File	The XSLT file to transform the request message before it is submitted to the native service. Click **Browse** to select a file from your file system and click **Save**. The XSL file uploaded by the user should not contain the XML declaration in it (`xml version="1.0" encoding="UTF-8"`). This is because when the virtual service is deployed to CloudStreams, CloudStreams embeds the XSL file in the virtual service definition (VSD), and since the VSD itself is in XML format, there cannot be an XML declaration line in the middle of it. This can lead to unexpected deployment issues which can be avoided by making sure the XSL file does not contain the declaration line. **Note:** If you make changes to the XSLT file in the future, you must re-deploy the virtual service.

5. To create an additional Transform step, right-click **In Sequence** and select **Transform** again.

6. To delete a Transform step, right-click **Transform** and click **Delete Step**.

The "Invoke IS Service" Step (Inbound, REST)

This optional step invokes a user-defined Integration Server flow service. You can invoke an IS service in:

- The "In Sequence" step, to pre-process the request message before it is submitted to the native service.

- The “Out Sequence” step, to pre-process the response message from the native service before it is returned to the consuming application.

The IS service you invoke must be running on the same Integration Server as CloudStreams. It can call out a C++ or Java or .NET function. It can also call other IS services to manipulate the SOAP message.

You can use predefined or custom context variables in an IS service. For more information, see “Using Context Variables in IS Services” on page 96.

You can create multiple “Invoke IS Service” steps.

To add the “Invoke IS Service” step (inbound, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Right-click **In Sequence** and click **Invoke IS Service**.

   The “Invoke IS Service” step is added under the Entry Step (and under a Transform step, if one exists). You cannot change the order of the steps.

4. Click **Invoke IS Service** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only.) **Invoke IS Service**.
Service	Specify the IS service to pre-process the request message before it is submitted to the native service.

5. When you define the IS service, the **Pipeline In** section of the flow should have the following input variables:

   - `proxy.name`: This is the name of the virtual service.
   - `SOAPEnvelope`: This is of the Java type org.apache.axiom.soap.SOAPEnvelope.
   - `MessageContext`: CloudStreams will automatically place a `MessageContext` variable into the pipeline before executing the IS service call. `MessageContext` is of the Java type org.apache.axis2.context.MessageContext.

Integration Server users can use the Axis2 `MessageContext` object to manipulate the incoming SOAP request. The Integration Server provides built-in services (the `pub.soap.*` services) to work with the `MessageContext` object to get/set/modify the SOAP body, header, properties, etc. Integration Server users should use these services to extract the information they need from the `MessageContext` to build the necessary business logic. Users do not need to understand Axis2 or Axiom (the xml object model based on StAX) to work with the SOAP request, because if they are familiar with the Integration Server pub.soap.* services, they can accomplish most of the tasks. For more information about these related Integration Server services, see the *webMethods Integration Server Built-In Services Reference*.

6. To create an additional Invoke IS Service step, right-click **In Sequence** and select **Invoke IS Service** again.

7. To delete an Invoke IS Service step, right-click **Invoke IS Service** and click **Delete**.

The "Routing Rule" Step (REST)

The Routing Rule step is required. You can choose one of the following protocols in your Routing Rule step for routing the requests:

- "Straight Through" routing (to route requests directly to the native service endpoint).
- "Context-Based" routing (to route specific types of messages to specific endpoints according to context-based routing rules).
- "Content-Based" routing (to route specific types of messages to specific endpoints based on specific values that appear in the request message).
- "Load Balancing" routing (to distribute requests among multiple endpoints).

The "Straight Through" Routing Rule Step (REST)

When you select the "Straight Through" routing protocol, the virtual service will route the requests directly to the native service endpoint you specify. You may specify how to authenticate requests (as with all routing protocols).

To configure the Routing Rule step for "Straight Through" routing (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Straight Through**.
Default To	The URL of the service to which to route the request. Click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows: ■ **HTTP Properties** ■ **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server. ■ **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds. ■ **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank. ■ **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication. ■ **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.
HTTP Method	The HTTP method to pass to the native service. Typically you want to pass each request to the native service with the same HTTP method that is contained in the request. For example, if a request contains a GET method, you allow the GET method to be passed to the native service. In this case, select the value **CUSTOM**, which is a context variable that contains the HTTP method that is contained in the request. However, in other cases you might want to change the HTTP method of a request to a different HTTP method. In this case, you

Option	Description
	can specify the different method explicitly (by selecting the GET, POST, PUT or DELETE value), or you can specify the different method dynamically on a per-request basis. If you want to specify the method dynamically, select the **CUSTOM** option and then write an IS service to set the value of the context variable. For more information, see "Changing the HTTP Method of a REST Virtual Service" on page 154.
Use credentials from incoming request	This is the default HTTP Authentication option. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use specific credentials	Authenticates requests according to the values you specify in the **User** and **Password** fields, and optionally the **Domain** field.
Invoke service anonymously	Does not authenticate requests.
Use existing HTTP Headers	Use the HTTP headers that are contained in the requests.
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on this page. If you need to specify multiple headers, use the plus button to add rows.

The "Context-Based" Routing Rule Step (REST)

If you have a native service that is hosted at two or more endpoints, you can use the Context-Based protocol to route specific types of messages to specific endpoints.

The requests are routed according to the context-based routing rules you create. A routing rule specifies where the requests should be routed, and the criteria by which they should be routed there. For example, requests can be routed according to certain consumers, certain dates/times, or according to requests that exceed/fall below a specified metric (Total Count, Success Count, Fault Count, etc.). You can create one or more rules. For example, you might use this capability to route requests from certain high-priority consumers to endpoints on a fast machine.

To configure the "Routing Rule" step for "Context-Based" routing (REST)

- Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
- In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.
- Expand **In Sequence**.

- Click **Routing Rule** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Context-Based**.
Rule Name	Assign a name for the rule. Then, click the icon next to this field and choose one of the following variables in the Configure Routing Rule dialog box that appears: **Time**, **IP Address** (IPv4 or IPv6 format), **Date**, **Consumer**, **Predefined Context Variable** or **Custom Context Variable** (see "Using Context Variables in IS Services" on page 96). Then specify a value and operator appropriate for your chosen variable. If you need to specify multiple variables, use the plus button at the end of the row to add rows. **Note:** If you select the value **Custom Context Variable**, you must write an IS service to get/set the custom context variable, and then invoke that service in an **Invoke IS Service** step. CloudStreams provides an API to get/set custom context variables. For more information, see "The API for Context Variables" on page 100. CloudStreams automatically declares any custom context variables you have specified; there is no need for you to declare them.
Route To	Specify the URL of the native service to route the request to, if the rule criteria are met. Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows: ■ **HTTP Properties** ■ **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server. ■ **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds.

Option	Description

	■ **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank.
	  ■ **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication.
	  ■ **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.

HTTP Method	The HTTP method to pass to the rule.
Default To	Enter a native service endpoint to route the request to in case all routing rules evaluate to False. Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box, as described for the **Route To** field above.
HTTP Method	The HTTP method to pass to the native service. Typically you want to pass each request to the native service with the same HTTP method that is contained in the request. For example, if a request contains a GET method, you allow the GET method to be passed to the native service. In this case, select the value **CUSTOM**, which is a context variable that contains the HTTP method that is contained in the request. However, in other cases you might want to change the HTTP method of a request to a different HTTP method. In this case, you can specify the different method explicitly (by selecting the GET, POST, PUT or DELETE value), or you can specify the different method dynamically on a per-request basis. If you want to specify the method dynamically, select the **CUSTOM** option and then write an IS service to set the value of the context variable. For more information, see "Changing the HTTP Method of a REST Virtual Service" on page 154.
Use credentials from incoming request	Default. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use specific credentials	Authenticates requests according to the values you specify in the **User**, **Password** and **Domain** fields.
Invoke service anonymously	Does not authenticate requests.

Option	Description
Use existing HTTP Headers	Use the HTTP headers that are contained in the requests.
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on this page. If you need to specify multiple headers, use the plus button to add rows.

The "Content-Based" Routing Step (REST)

If you have a native service that is hosted at two or more endpoints, you can use the Content-Based routing protocol to route specific types of messages to specific endpoints based on specific values that appear in the request message.

You might use this capability, for example, to determine which operation the consuming application has requested, and route requests for complex operations to an endpoint on a fast machine.

The requests are routed according to the content-based routing rules you create. That is, they are routed based on the successful evaluation of one or more XPath expressions that are constructed utilizing the content of the request payload. For example, a routing rule might allow requests for half of the methods of a particular service to be routed to Service A, and the remaining methods to be routed to Service B.

To configure the "Routing Rule" step for Content-Based routing (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Content-Based**.
Rule Name	Assign a name to the rule.

Option	Description
XPath	The XPath field is applicable only while creating SOAP Virtual Services.

Route To	Specify where to route the request if the rule criteria are met. Specify either the URL of a native service or a connection pool name.
	
	Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows:
	
	■ **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server.
	
	■ **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds.
	
	■ **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank.
	
	  ■ **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication.
	
	  ■ **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.

HTTP Method	The HTTP method to pass to the rule.
Default To	Enter a native service endpoint to route the request to in case all routing rules evaluate to False. Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box, as described for the **Route To** field above.
HTTP Method	The HTTP method to pass to the native service. Typically you want to pass each request to the native service with the same HTTP method that is contained in the request. For example, if a request contains a GET method, you allow the GET method to be passed to the native service. In this case, select the value **CUSTOM**, which is a context variable that contains the HTTP method that is contained in the request. However, in other cases you might want to change the HTTP method of a request to a different HTTP method. In this case, you

Option	Description
	can specify the different method explicitly (by selecting the GET, POST, PUT or DELETE value), or you can specify the different method dynamically on a per-request basis. If you want to specify the method dynamically, select the **CUSTOM** option and then write an IS service to set the value of the context variable. For more information, see "Changing the HTTP Method of a REST Virtual Service" on page 154.
Use credentials from incoming request	Default. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use specific credentials	Authenticates requests according to the values you specify in the **User**, **Password** and **Domain** fields.
Invoke service anonymously	Does not authenticate requests.
Use existing HTTP Headers	Use the HTTP headers that are contained in the requests.
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on this page. If you need to specify multiple headers, use the plus button to add rows.

The "Load Balancing" Routing Rule Step (REST)

If you have a Web service that is hosted at two or more endpoints, you can use the Load Balancing option to distribute requests among the endpoints.

The requests are intelligently routed based on the "round-robin" execution strategy. The load for a service is balanced by directing requests to two or more services in a pool, until the optimum level is achieved. The application routes requests to services in the pool sequentially, starting from the first to the last service, without considering the individual performance of the services. After the requests have been forwarded to all the services in the pool, the first service is chosen for the next loop of forwarding.

Load-balanced endpoints also have automatic Failover capability. If a load-balanced endpoint is unavailable (for example, if a connection is refused), then that endpoint is marked as "down" for the number of seconds you specify in the **Suspend Interval** field (during which the endpoint will not be used for sending the request), and the next configured endpoint is tried. If all the configured load-balanced endpoints are down, then a SOAP fault is sent back to the client. After the suspension period expires, each endpoint marked will be available again to send the request.

To configure the "Routing Rule" step for Load Balancing routing (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.

3. Expand **In Sequence**.

4. Click **Routing Rule** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **HTTP**.
Routing Type	Select **Load Balancing**.
Route To	The URLs of two or more services in a pool to which the requests will be routed. The application routes the requests to services in the pool sequentially, starting from the first to the last service without considering the individual performance of the services. After the requests have been forwarded to all the services in the pool, the first service is chosen for the next loop of forwarding. To specify additional services, use the plus button next to the field to add rows. Then, click the icon next to this field and complete the Configure Endpoint Properties dialog box as follows. These properties will apply to all the endpoints. ■ **HTTP Properties** ■ **Connection Timeout**: The time interval (in seconds) after which a connection attempt will timeout. If a value is not specified (or if the value 0 is specified), CloudStreams will use the default value specified in Integration Server. ■ **Read Timeout**: The time interval (in seconds) after which a socket read attempt will timeout. If a value is not specified (or if the value 0 is specified), the default is 30 seconds. ■ **SSL Options**: Optional. To enable SSL client authentication for the endpoint, you must specify values for both the **Client Certificate Alias** field and the **IS Keystore Alias** field. If you

Option	Description
	specify a value for only one of these fields, a deployment error will occur. You may leave both fields blank. ■ **Client Certificate Alias**: The client's private key to be used for performing SSL client authentication. ■ **IS Keystore Alias**: The keystore alias of the instance of Integration Server on which CloudStreams is running. This value (along with the value of **Client Certificate Alias**) will be used for performing SSL client authentication.
Suspend Interval	A numeric timeout value (in seconds) for a failed endpoint. Default: 30. When this timeout value expires, the system routes the execution of the virtual service to the next consecutive Web service endpoint specified in the **Route To** field.
HTTP Method	The HTTP method to pass to the native service. Typically you want to pass each request to the native service with the same HTTP method that is contained in the request. For example, if a request contains a GET method, you allow the GET method to be passed to the native service. In this case, select the value **CUSTOM**, which is a context variable that contains the HTTP method that is contained in the request. However, in other cases you might want to change the HTTP method of a request to a different HTTP method. In this case, you can specify the different method explicitly (by selecting the GET, POST, PUT or DELETE value), or you can specify the different method dynamically on a per-request basis. If you want to specify the method dynamically, select the **CUSTOM** option and then write an IS service to set the value of the context variable. For more information, see "Changing the HTTP Method of a REST Virtual Service" on page 154.
Use Credentials from Incoming Request	This is the default HTTP Authentication option. Authenticates requests based on the credentials specified in the HTTP header. CloudStreams passes the `Authorization` header present in the original client request to the native service.
Use Specific Credentials	Authenticates requests according to the values you specify in the **User**, **Password** and **Domain** fields.
Invoke Service Anonymously	Does not authenticate requests.
Use Existing HTTP Headers	Use the HTTP headers that are contained in the requests.

Option	Description
Customize HTTP Headers	Use the HTTP headers that you specify in the **Name** and **Value** columns on the tab. If you need to specify multiple headers, use the plus button to add rows.

The "Out Sequence" Step (REST)

You configure the service's "Out Sequence" step (optional) to manipulate the response messages from the native service before they are returned to the consuming applications. This step can include:

- The "Transform" step (optional), which invokes an XSLT transformation file to transform the SOAP response payloads from XML format to the format required by the consumer.
- The "Invoke IS Service" step (optional), which processes the response message from the native service provider before it is returned to the consuming application.

The "Transform" Step (Outbound, REST)

Add this step if you need to invoke an XSLT transformation file to transform the native service's response messages into the format required by the consumer applications, before CloudStreams returns the responses to the consumer applications. For more information about when to use the Transform step, see "The Transform Step (Inbound, REST)" on page 127.

To add the "Transform" step (outbound, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the virtual service name.
3. In the virtual service editor, expand **Out Sequence**.
4. To add the Transform step, right-click **Out Sequence** and select **Transform**.

   The **Transform** step is added under the **Out Sequence** step.
5. Click **Transform** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.

Option	Description
XSLT File	The XSLT file to transform the response message before it is returned to the consuming application. Click **Browse** to select a file from your file system and click **Save**. The XSL file uploaded by the user should not contain the XML declaration in it (`xml version="1.0" encoding="UTF-8"`). This is because when the virtual service is deployed to CloudStreams, CloudStreams embeds the XSL file in the virtual service definition (VSD), and since the VSD itself is in XML format, there cannot be an XML declaration line in the middle of it. This can lead to unexpected deployment issues which can be avoided by making sure the XSL file does not contain the declaration line. **Note:** If you make changes to the XSLT file in the future, you must re-deploy the virtual service.

6. To create an additional Transform step, right-click **In Sequence** and select **Transform** from the context menu.

7. To delete a Transform step, right-click **Transform** and click **Delete**.

The "Invoke IS Service" Step (Outbound, REST)

This optional step invokes a user-defined Integration Server flow service. You can invoke an IS service in the service's "Out Sequence" step to pre-process the response message from the native service before it is returned to the consuming application. For more information about IS services, see "The Invoke IS Service Step (Inbound, REST)" on page 128.

To add the "Invoke IS Service" step (outbound, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Right-click **Out Sequence** and click **Invoke IS Service**.

   The Invoke IS Service step is added to the Out Sequence step.

4. In the Properties view, complete the following fields.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Invoke IS Service**.
Service	Specify the IS Service to pre-process the response message before it is returned to the consuming application.

5. When you define the IS service, the **Pipeline In** section of the flow should have the following input variables:

   - `proxy.name`: This is the name of the virtual service.
   - `SOAPEnvelope`: This is of the Java type org.apache.axiom.soap.SOAPEnvelope.
   - `MessageContext`: CloudStreams will automatically place a `MessageContext` variable into the pipeline before executing the IS service call. `MessageContext` is of the Java type org.apache.axis2.context.MessageContext.

     Integration Server users can use the Axis2 `MessageContext` object to manipulate the incoming SOAP request. The Integration Server provides built-in services (the pub.soap.* services) to work with the `MessageContext` object to get/set/modify the SOAP body, header, properties, etc. Integration Server users should use these services to extract the information they need from the `MessageContext` to build the necessary business logic. Users do not need to understand Axis2 or Axiom (the xml object model based on StAX) to work with the SOAP request, because if they are familiar with the Integration Server pub.soap.* services, they can accomplish most of the tasks. For more information about these related Integration Server services, see the *webMethods Integration Server Built-In Services Reference*.

6. To create an additional Invoke IS Service step, right-click **In Sequence** and click **IS Service** again.

7. To delete an Invoke IS Service step, right-click **Invoke IS Service** and click **Delete**.

The "Error Sequence" Step (REST)

CloudStreams returns a default fault response to the consuming application, which you can customize with context variables. This fault response is used for faults returned by the native service provider as well as faults returned by internal CloudStreams exceptions, such as policy violation errors, connection timeouts, etc. In addition, you can:

- Choose whether or not to send the native service provider's service fault content, or just send the response message.
- Invoke IS services to pre-process or post-process the error messages.

You use this step to configure error messaging for each virtual service individually. If you want to configure global error responses for *all* virtual services, set the Service Fault Configuration

options, which are located in the Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Service Fault Configuration**). For details, see "Setting the Service Fault Configuration Options" on page 39.

To configure the Error Sequence step (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name.

3. Expand **Error Sequence**.

4. Click **Error Messaging** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Error Messaging**.
Error Message	Select one or both of the following options: **Custom Failure Response Message**: When you select this option, CloudStreams returns the following fault response to the consuming application: `CloudStreams encountered an error:$ERROR_MESSAGE while executing operation:$OPERATION service:$SERVICE at time:$TIME on date:$DATE. The client ip was:$CLIENT_IP. The current user:$USER. The consumer application:$CONSUMER_APPLICATION".` **Note:** When $CLIENT_IP is used, CloudStreams will replace $CLIENT_IP with the IP address of the client. For privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), delete the *"The client ip was:$CLIENT_IP."* string. This fault response is returned in both of the following cases: ■ When a fault is returned by the native service provider. In this case, the `$ERROR_MESSAGE` variable in the fault response will contain the message produced by the provider's exception that caused the error. This is equivalent to the `getMessage` call

Option	Description

on the Java Exception. For REST service calls, the message is returned inside an `</Exception>` tag. If the response is XML, the message is returned inside `<Exception>'custom message'</Exception>`. If the response is JSON, it will be returned inside `{"Exception":"Invalid response"}`. CloudStreams discards the native service provider's fault and does not return this content to the web service caller since it could be considered a security issue, especially if the native provider is returning a stack trace with its response.

- When a fault is returned by internal CloudStreams exceptions (policy violation errors, cloud connection errors and cloud connector service errors).

  In this case, the `$ERROR_MESSAGE` variable will contain errors generated by CloudStreams.

The default fault response contains predefined fault handler variables (`$ERROR_MESSAGE`, `$OPERATION`, etc.), which are described in "The Fault Handler Variables" on page 87.

You can customize the default fault response using the following substitution variables, where CloudStreams replaces the variable reference with the real content at run time:

- The predefined context variables listed in "The Predefined Context Variables" on page 97.
- Custom context variables that you declare using the CloudStreams API (see "The API for Context Variables" on page 100).

Note:
If you want to reference a custom context variable that you have already defined in a context-based routing rule (as opposed to one you have declared using CloudStreams API for context variables), then you must add the prefix `$mx` to the variable name in order to reference the variable. For example, if you defined the variable `TAXID`, you would reference it as `$mx:TAXID`.

Native Provider Fault: When you select this option, CloudStreams sends the native service provider's service fault content, if one is available. CloudStreams will send whatever content it received from the native service provider.

If you select this option, the **Custom Failure Response Message** is ignored when a fault is returned by the native service provider. (Faults returned by internal CloudStreams exceptions will still be handled by the **Custom Failure Response Message** option.)

Option	Description
Processing Method	Optionally select either of the following: ■ **Pre-Processing**: Select this option if you want to invoke an IS service to manipulate the response message before the Error Sequence step is invoked. The IS service will have access to the response message context (the axis2 `MessageContext` instance) before it is updated with the custom error message. For example, you might want to send emails or perform custom alerts based on the response payload. For more information about IS services, see "The Invoke IS Service Step (Inbound, REST)" on page 128. ■ **Post-Processing**: Select this option if you want to invoke an IS service to manipulate the service fault after the Error Sequence step is invoked. The IS service will have access to the entire service fault and the custom error message. You can make further changes to the fault message structure, if needed. For more information about IS services, see "The Invoke IS Service Step (Inbound, REST)" on page 128.

Creating a New Connector Virtual Service (REST)

A connector virtual service will:

- Handle the provider's response.
- Log the requests/responses.

CloudStreams provides a default connector virtual service for each metadata handler: the service `WmCloudStreams.RestVS` (for the REST handler) and the service `WmCloudStreams.SoapVS` (for the SOAP handler). The default services are located in the sample CloudStreams Governance project in the WmCloudStreams package. You cannot modify these default services.

Each default connector virtual service has a default policy (named `Logging Policy`), which logs all request/response payloads to a database. You cannot modify the default policy. Alternatively, you can create additional connector virtual services with custom policies. If you create a connector virtual service with a custom policy, you can only include the actions in the "Monitoring" or "Additional" action categories; you cannot include the "WS-SecurityPolicy 1.2" actions. For example, you might want to create a custom policy that monitors run-time performance, customizes how the service invocations are logged, validates response messages against an XML schema, or optimizes server traffic. For complete details, see the chapter "Policies" on page 159.

To create a new connector virtual service (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, right-click the CloudStreams Governance project and click **New > Connector Virtual Service** (or expand the project, right-click the **Connector Virtual Services** folder and click **New Connector Virtual Service**).

3. In the New Connector Virtual Service wizard, complete the following fields and click **Finish**.

Option	Description
Project	Click **Browse** and select a CloudStreams Governance project in which to create the connector virtual service.
Name	Assign a name for the service. Unlike native services, the names of virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the virtual service name, then the names of the services themselves must not contain characters that are invalid in virtual service names. **Note:** If you want to change the service name after it has been created, right-click the service name in the **Connector Virtual Services** folder and select **Rename**.
Version	The version is always set to 1.0.
Service Type	(Read-only field.) **REST**.
Description	Optional. A description for the service. This description appears when a user displays a list of services in the user interface.

The new connector virtual service is added to the CloudStreams Governance project, in the **Connector Virtual Services** folder.

The Properties of a Connector Virtual Service (REST)

To view the properties of a connector virtual service (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the virtual service name. (If the Properties view is not already open, click **Window > Show View > Other > General > Properties**.)

3. The **General** page in the Properties view displays the following properties:

Option	Description
Name	(Read-only field.) The service name. You can change the service name at any time by right-clicking the name in the **Virtual Services** folder and clicking **Rename**.
Service Type	(Read-only field.) **REST**.
Created/Last Modified	(Read-only field.) The service's creation/modification timestamps.
Target Namespace	(Read-only field.) The value is derived from the `targetNamespace` attributes of the WSDL's definition element.
Namespaces	Click the button next to this field to view the service's available namespaces (such as `wsdl`, `xsd`, etc.).
Version	The version is always set to 1.0.
WSDL URL	(Read-only field.) The URL of the service's WSDL.
Description	You can change the service description.

4. View additional properties in the **Advanced** page.

Option	Description
Name	(Read-only field.) The service name.
Type	(Read-only field.) The type of the service (REST).
Target Namespace	(Read-only field.) The value derived from the `targetNamespace` attributes of the WSDL's definition element.
WSDL URL	Click this URL to display the contents of the service's abstract WSDL. If a WSDL file was not added, this will be empty.
Namespaces	Click the button next to this field to view the service's available namespaces (such as `wsdl`, `xsd`, etc.).
Version	The version is always set to 1.0.
Description	(Read-only field.) The service description.
VSD	Click the button next to this field to view the service's “virtual service definition” (VSD). This button is disabled until you deploy the connector virtual service to a CloudStreams server. When you deploy the connector virtual service to a CloudStreams server, CloudStreams generates an XML document called a *virtual service definition (VSD)*. The VSD defines the connector virtual service for CloudStreams, and contains all the resources required

Option	Description
	to deploy the connector virtual service to a CloudStreams server, including the policy that applies to the service. You cannot edit the VSD. **Note:** If multiple policies apply to the service, CloudStreams combines all those policies into a single policy known as the *effective policy*. The effective policy is a simple UNION of the run-time actions specified in all policies that apply to a service. To create the effective policy, CloudStreams evaluates the combined list of actions from all policies, using a set of internal rules known as Policy Conflict Resolution rules. For details, see "Policy Conflict Resolution Rules" on page 195.
Applicable Policies	Click the button next to this field to view a list of the active policies that apply to this service. Any inactive policy that applies to the service is not listed.
Endpoint	(Read-only field). The endpoint is resolved when the connector service is configured and the user selects an enabled managed connection pool.
Deployed Status	(Read-only field.) Indicates whether the service is **Deployed**, **Undeployed** or **Not Deployed** (which is the initial status before you deploy the service). To deploy or undeploy a service, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Resource	Click the button next to this field to view the REST resources that the service will access.

Managing a Connector Virtual Service (REST)

You can rename, delete, deploy or undeploy a connector virtual service, and change the list of policies that apply to it.

To manage a connector virtual service (REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project.

3. Right-click the connector virtual service name and choose any of the following tasks from the context menu:

Option	Description
Rename	Use this option to assign a new name to the service. The service must be undeployed before you can rename it (check the **Deployed Status** field in the Advanced page of the service's Properties view). Unlike native services, the names of connector virtual services cannot contain spaces or special characters except _ (underscore) and - (hyphen). Consequently, if you adopt a convention that involves using the name of the service as part of the connector virtual service name, then the names of the services themselves must not contain characters that are invalid in connector virtual service names.
Delete	Deletes the connector virtual service from CloudStreams.
Deploy	Deploys the connector virtual service to a CloudStreams server target. For more information, see "Deploying Virtual Services and Connector Virtual Services" on page 191.
Undeploy	Undeploys the connector virtual service from a CloudStreams server target.

4. To change the list of policies that apply to the connector virtual service, see "Modifying Policies" on page 164.

The "In Sequence" Step (Connector Virtual Service, REST)

The "In Sequence" step can contain the following sub-steps:

- The "Entry Step" (required), which specifies the protocol of the requests that the connector virtual service will accept, and the HTTP methods that the connector virtual service should be allowed to perform on a REST resource.
- The "Routing Rule" step (required), which specifies how the connector virtual service will route the service requests to the native service endpoint.
- The "Invoke IS Service" step (optional), which pre-processes the request message before it is submitted to the native service.

The "Entry Step" (Connector Virtual Service, REST)

This step specifies the protocol of the requests that the connector virtual service will accept, and the HTTP methods that the virtual services are allowed to perform on the REST resources.

To edit the "Entry Step" (connector virtual service, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the connector virtual service name.

   The virtual service editor displays the three main processing steps: **In Sequence**, **Out Sequence** and **Error Sequence**.

3. Expand **In Sequence**.

   By default, the **In Sequence** step contains the sub-steps **Entry Step** and **Routing Rule**. (You can add the optional **Invoke IS Service** step, as described later.)

4. Click **Entry Step** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Entry Step**.
Protocol	(Read-only field.) The protocol of the requests that the connector virtual service will accept. The value Local means that the service can only be called from Cloud Connector Services.
HTTP Method	The HTTP methods that the virtual services should be allowed to perform on the REST resources (GET, POST, PUT, DELETE). It is important to specify all the HTTP methods that are supported for the virtual services. For example, if a virtual service is deployed to CloudStreams and only the GET method was selected here, then CloudStreams will only permit GET invocations. A POST request will be rejected with a return of statusCode 405 even if the native service happens to support POSTs. It is also important for the native service to return the correct `Content-Type` header with its responses. At run time, CloudStreams determines the type of a request message based on the message's HTTP method and its Content-Type. For a list of valid HTTP method/Content-Type combinations, see "HTTP Method/Content-Type Combinations" on page 153.

The "Routing Rule" Step (Connector Virtual Service, REST)

This step specifies how the connector virtual service will route the requests to the native service endpoint. You cannot modify this step, except for its name.

To edit the Routing Rule step (connector virtual service, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and double-click the connector virtual service name.

3. In the virtual service editor, expand **In Sequence**.

4. Click **Routing Rule** and complete the following fields in the **General** page in the Properties view.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Routing Rule**.
Protocol	(Read-only field.) **Protocol**. You cannot change the protocol over which the connector virtual service will accept requests.
Routing Rule	(Read-only field.) The value **Connection Pool** means that the requests are sent to the provider via the CloudStreams connection pool. For more information about connection pools, see the documentation specific to your CloudStreams connector (for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*).

The "Invoke IS Service" Step (Inbound, Connector Virtual Service, REST)

This optional step invokes a user-defined Integration Server flow service. An IS service is a user-defined Integration Server flow service that you can invoke in:

- The "In Sequence" step, to pre-process the request messages before CloudStreams submits the requests to the native service.
- The "Out Sequence" step, to pre-process the response messages from the native service before CloudStreams returns the responses to the consuming application.

You create an "Invoke IS Service" step for a connector virtual service the same way you create one for a virtual service. For details, see "The Invoke IS Service Step (Inbound, REST)" on page 128, or "The Invoke IS Service Step (Outbound, REST)" on page 141.

The "Out Sequence" Step (Connector Virtual Service, REST)

You configure the service's "Out Sequence" step (optional) to manipulate the response messages from the native service before CloudStreams returns the responses to the consuming applications. This step can include the "Invoke IS Service" step (optional), which processes the response message from the native service provider before it is returned to the consuming application. See "The Invoke IS Service Step (Inbound, Connector Virtual Service, REST)" on page 151.

The "Error Sequence" Step (Connector Virtual Service, REST)

The default connector virtual services are configured to return the native service provider's service fault, if one is available. CloudStreams will send whatever content it received from the native service provider. You can optionally invoke IS services to pre-process or post-process the error messages.

To configure the Error Sequence step (connector virtual service, REST)

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.

2. In the CloudStreams Governance view, expand your CloudStreams Governance project and click the connector virtual service name.

3. Expand **Error Sequence**.

4. Click **Error Messaging** and complete the fields in the **General** page in the Properties view as follows.

Option	Description
Name	You can optionally change the step name to any other name. There are no naming restrictions.
Type	(Read-only field.) **Error Messaging**.
Processing Method	Optionally select either of the following: ■ **Pre-Processing**: Select this option if you want to invoke an IS service to manipulate the response message before the Error Sequence step is invoked. The IS service will have access to the response message context (the axis2 `MessageContext` instance) before it is updated with the custom error message. For example, you might want to send emails or perform custom alerts based on the response payload. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69. ■ **Post-Processing**: Select this option if you want to invoke an IS service to manipulate the service fault after the Error Sequence step is invoked. The IS service will have access to the entire service fault and the custom error message. You can make further changes to the fault message structure, if needed. For more information about IS services, see "The Invoke IS Service Step (Inbound, SOAP)" on page 69.

Important Considerations for REST Virtual Services

Keep the following topics in mind when you configure REST virtual services:

- "HTTP Method/Content-Type Combinations" on page 153
- "Changing the HTTP Method of a REST Virtual Service" on page 154

HTTP Method/Content-Type Combinations

When you configure a REST virtual service it is important to specify all the HTTP methods that are supported for the service. For example, if the REST virtual service is deployed to CloudStreams and only the GET method was selected in the REST virtual service definition, then CloudStreams will only permit GET invocations. A POST request will be rejected with a return of statusCode 405 even if the native service happens to support POSTs.

At run time, CloudStreams determines the type of a request message based on the message's HTTP method and its Content-Type. (The absence of the `soapAction` header will indicate to CloudStreams that the message is an XML message.)

The valid HTTP method/Content-Type combinations are as follows:

This method...	Can be included in a message of this type...
POST	application/xml
	application/json
	application-x-www-form-urlencoded
	multipart/form-data
PUT	application/xml
	application/json
	application-x-www-form-urlencoded
	multipart/form-data
GET	application-x-www-form-urlencoded
DELETE	application-x-www-form-urlencoded

Note that:

- If CloudStreams receives a request sent with an HTTP method that is not specified in the REST virtual service definition, it will return a 405 error.
- If CloudStreams receives a request sent with a wrong Content-Type, it will return a 415 error. In addition, if the wrong Content-Type is used with a GET or DELETE, then the query parameters contained in the message (if any) will not be processed.

Changing the HTTP Method of a REST Virtual Service

When configuring the routing step of a REST virtual service, you specify whether to route the requests to the native service with the same HTTP method that is contained in the requests (GET, POST, PUT, DELETE), or whether to route the requests with a different HTTP method. This applies only to virtual services, not to connector virtual services.

Typically you want to pass each request to the native service with the same HTTP method that is contained in the request. For example, if a request contains a GET method, you allow the GET method to be passed to the native service. However, there might be rare cases in which you want to change the HTTP method of a request to a different HTTP method. For example, you might want to:

- Expose an XML service as a REST service.

  In this case, the service you create would be a Virtual XML service that exposes the HTTP methods GET, POST, PUT and DELETE, but the routing method would always be POST.

- Expose a REST service whose virtual REST service only exposes the POST method

To change the HTTP method of a REST request, in the REST virtual service's "Routing Rule" step, set the value of the **HTTP Method** field either statically (by explicitly setting the value to **GET**, **POST**, **PUT** or **DELETE**) or dynamically (by setting the value to **CUSTOM**).

In order to use the **CUSTOM** option to set the field dynamically, you must write a IS service that sets a value of GET, POST, PUT or DELETE for a predefined context variable named ROUTING_METHOD. You need to invoke this service in the virtual service's "In Sequence" step. For details, see "Changing HTTP Methods in Requests Dynamically Using a Context Variable" on page 156.

Note:
Use this feature carefully, since changing HTTP methods to certain other HTTP methods could result in unintended results or errors. For example, changing an inbound GET request to a DELETE request would be a serious mistake if the deletion was not intended and the native REST service actually deleted a resource when invoked with a DELETE method. Additionally, an incoming POST or PUT request cannot be translated into a GET or DELETE if the request has nested elements. For more information, see "Implications of Changing HTTP Methods" on page 154.

Implications of Changing HTTP Methods

Note the following.

When changing from...	To...	Note that...
GET	POST	■ The Content-Type of the changed request is sent as application/xml or application/json, and the charset is UTF-8.

When changing from...	To...	Note that...
		■ Depending on the structure of the native service, be aware that the native service might not be expecting the same payload structure that is being sent. In this case, you would need to transform the request message into the format required by the native service before CloudStreams sends the requests to the native service. For more information, see "Sample XSLT Transformation for GET-to-POST or GET-to-PUT" on page 157.
GET	PUT	Identical to GET-to-POST, except that CloudStreams changes the request's HTTP method from GET to PUT.
GET	DELETE	No comment.

POST	GET	■ CloudStreams will translate the POSTed request elements into query string parameters, in a root element. An incoming POST or PUT request cannot be translated into a GET or DELETE if the request has nested elements. For example:
		`(this is correct)`
		`<person>`
		`  <lastName>Smith</lastName>`
		`</person>`
		`(this is incorrect)`
		`<person>`
		`  <name>`
		`  <last>Smith</last>`
		`  </name>`
		`</person>`
		■ If you want to send additional parameters as part of the request URL, you can transform this payload. To do this, you can use an XSLT file or a webMethods IS service call to add parameters before the request is sent to the native service. For more information, see "The In Sequence Step (REST)" on page 125.

POST	DELETE	Identical to POST-to-GET, except that CloudStreams changes the request's HTTP method from POST to DELETE.
POST	PUT	The Content-Type of the changed request is sent as application/xml or application/json, and the charset is UTF-8.
PUT	GET	Identical to POST-to-GET, except that CloudStreams changes the request's HTTP method from PUT to GET.
PUT	POST	The Content-Type of the changed request is sent as application/xml or application/json, and the charset is UTF-8.

When changing from...	To...	Note that...
PUT	DELETE	Identical to POST-to-DELETE, except that CloudStreams changes the request's HTTP method from PUT to DELETE.
DELETE	GET	No comment.
DELETE	POST	Identical to GET-to-POST, except that CloudStreams changes the request's HTTP method from DELETE to POST.
DELETE	PUT	Identical to GET-to-PUT, except that CloudStreams changes the request's HTTP method from DELETE to PUT.
GET, POST, PUT or DELETE	CUSTOM	See "Changing HTTP Methods in Requests Dynamically Using a Context Variable" on page 156.
GET or DELETE	POST or PUT	Note that the query parameters will be picked off the URL and stored as top-level elements when the message is sent to the native service. The query parameters are ignored on the endpoint URL and lost when we POST to the native provider (that is, do not change the protocol method).

Changing HTTP Methods in Requests Dynamically Using a Context Variable

Alternatively, instead of changing an HTTP method explicitly (statically) to PUT, POST, GET or DELETE, you can change the HTTP method to the value of a predefined context variable (ROUTING_METHOD) that dynamically resolves to a different HTTP method (PUT, POST, GET or DELETE, as appropriate).

To change the HTTP method dynamically, you create an IS service and invoke it in the virtual service's "In Sequence" step. This IS service should reference the predefined context variable ROUTING_METHOD (see "The Predefined Context Variables" on page 97). To set the value of ROUTING_METHOD, use the setContextVariableValue method, which is defined in the following class:

com/softwareag/cloudstreams/api/RuntimeFacade.java

For example:

```
public static final void updateHttpMethod(IData pipeline)
   throws ServiceException {

String mcKey = "Message Context";

 Object obj = IDataUtil.get(pipeline.getCursor(), mcKey);
 if (obj!=null && obj instanceof
org.apache.axis2.context.MessageContext) {

  MessageContext msgCtx = (MessageContext) obj;

QName varName =
new QName(ContextVariableType.ROUTING_METHOD.getName());
```

```
RuntimeFacade.setContextVariableValue(varName, "PUT", msgCtx );
 }
}
```

Sample XSLT Transformation for GET-to-POST or GET-to-PUT

As stated in the above table, depending on the structure of the native service, the native service might not be expecting the same payload structure that is being sent. In this case, you would need to transform the request message into the format required by the native service before CloudStreams sends the requests to the native service. To do this, you invoke an XSLT file during the "In Sequence" step.

Assume that:

- The native service name is `"authors:`.
- The REST virtual service for `"authors:` is named `"vs-authors:` and is made available in CloudStreams at this endpoint: `http://localhost:5555/ws/vs-authors/Invoke`. The `targetNamespace` of the REST virtual service is `"http://example.com/authors"`.

Following is a sample XSLT transformation file for the GET-to-POST or GET-to-PUT scenario.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ns="http://example.com/authors"
    version="1.0">

    <xsl:output method="xml" omit-xml-declaration="no" standalone="yes"
indent="yes"/>

    <xsl:strip-space elements="*"/>
    <xsl:template match="node()|@*">
        <xsl:copy>
            <xsl:apply-templates select="node()|@*"/>
        </xsl:copy>
    </xsl:template>

    <xsl:template match="//ns:invoke/node()">
        <xsl:element name="{local-name(.)}">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="//ns:invoke">
        <xsl:element name="authors">
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

4 Policies

Overview of Policies

A policy adds run-time governance capabilities to a virtual service or a Connector Virtual Service. A *policy* is a sequence of actions that are carried out by CloudStreams when a consumer requests a particular service through CloudStreams.

The actions in a policy perform activities such as identifying/authenticating consumers, validating digital signatures and capturing performance measurements. An *action* is a single task that is included in a policy and is evaluated by CloudStreams at run time. Actions have one or more parameters, which you configure when you insert the actions into a policy. For example, an action that identifies consumers specifies one or more identifiers to identify the consumers who are trying to access the services.

CloudStreams provides built-in action templates. A built-in action template is a definition of an action that can be used in a policy. An action template specifies the set of parameters associated with a particular policy action. You can use these action templates to create actions for policies.

When you create a policy, you:

1. Specify the service(s) to which the policy should apply. A policy can apply to one or more virtual services or one or more Connector Virtual Services, but not to both types of services.
2. Add the desired actions to the policy, and configure their parameters.
3. Activate the policy when you are ready to put it into effect. When you deploy the virtual services or Connector Virtual Services to which the policy is applied, the policy will also be deployed.

Following is a summary of the kinds of actions you can include in policies for virtual services and Connector Virtual Services. Detailed descriptions of the actions appear later, in "The Policy Actions" on page 165.

Policies for Virtual Services

You should create a policy for each virtual service. You can apply a single policy to one or more virtual services. A policy for a virtual service can include the following kinds of actions:

WS-SecurityPolicy 1.2 Actions

There are two kinds of WS-SecurityPolicy 1.2 actions:

- Authentication actions, to verify that the requests for virtual services contain a specified WS-SecurityPolicy element. CloudStreams provides the following authentication actions:
  - **Require SAML Token**: Requires that a WSS Security Assertion Markup Language (SAML) assertion token be present in the SOAP message header to validate service consumers.
  - **Require WSS Username Token**: Requires that a WSS username token and password be present in the SOAP message header to validate service consumers.

  - **Require X.509 Token**: Requires that a WSS X.509 token be present in the SOAP message header to validate service consumers.

- XML security actions, to provide confidentiality (through encryption) and integrity (through signatures) for request and response messages. CloudStreams provides the following XML security actions:

  - **Require Signing**: Requires that a request's XML element (which is represented by an XPath expression) be signed.

  - **Require Encryption**: Requires that a request's XML element (which is represented by an XPath expression) be encrypted.

  - **Require SSL**: Requires that requests be sent via SSL client certificates, and can be used with both SOAP and REST services.

  - **Include Timestamps**: Requires that timestamps be included in the request header. CloudStreams checks the timestamp value against the current time to ensure that the request is not an old message. This serves to protect your system against attempts at message tampering, such as replay attacks.

Monitoring Actions

CloudStreams provides the following run-time monitoring actions:

- **Monitor Service Performance**: This action monitors a user-specified set of run-time performance conditions for a virtual service, and sends alerts to a specified destination when these performance conditions are violated.

- **Monitor Service Level Agreement**: This action provides the same functionality as "Monitor Service Performance", but this action is different because it enables you to monitor a virtual service's run-time performance for particular consumers. You can configure this action to define a *Service Level Agreement* (SLA), which is set of conditions that defines the level of performance that a specified consumer should expect from a service.

- **Throttling Traffic Optimization**: This action limits the number of service invocations allowed during a specified time interval, and sends alerts to a specified destination when the performance conditions are violated. You can use this action to avoid overloading the back-end services and their infrastructure, to limit specific consumers in terms of resource usage, etc.

Additional Actions

CloudStreams provides additional actions that you can use in conjunction with the actions above.

- **Identify Consumer**: You use this action in conjunction with an authentication action ("Require WSS Username Token", "Require X.509 Token", or "Require HTTP Basic Authentication"). Alternatively, this action can be used alone to identify consumers only by host name or IP address.

- **Require HTTP Basic Authentication**: This action uses HTTP Basic authentication to verify the consumer's authentication credentials contained in the request's Authorization header.

CloudStreams authorizes the credentials against the list of users registered in the Integration Server on which CloudStreams is running. This action supports WS-SecurityPolicy 1.2.

- **Authorize User**: This action authorizes consumers against a list of users and/or a list of groups registered in the Integration Server on which CloudStreams is running. You use this action in conjunction with an authentication action ("Require WSS Username Token", "Require SAML Token", or "Require HTTP Basic Authentication").
- **Log Invocation**: Logs request/response information (and optionally their payloads) to a destination you specify. See "Log Invocation" on page 171 for General Data Protection Regulation (GDPR) related information.
- **Validate Schema**: Validates all XML request and/or response messages against an XML schema referenced in the WSDL.

Policies for Connector Virtual Services

Each default Connector Virtual Service has a default policy, which logs the requests/responses to a database. You cannot modify the default Connector Virtual Service or its policy. However, you can create additional Connector Virtual Services with custom policies. If you create a Connector Virtual Service with a custom policy, you can include the actions from the "Monitoring" or "Additional" action categories (except for "Validate Schema"); the "WS-SecurityPolicy 1.2" actions are not needed. For example, you might want to create a custom policy that monitors run-time performance, customizes how the service invocations are logged, or optimizes server traffic.

The default Connector Virtual Service policies are named `Logging Policy`, located in the sample CloudStreams Governance project in the WmCloudStreams package.

The remainder of this section describes:

Creating Policies

To create a policy

1. Open Software AG Designer and display the CloudStreams Development perspective.
2. In the CloudStreams Governance view, expand your CloudStreams Governance project folder, right-click the **Policies** folder and click **New Policy**.
3. In the New Policy wizard, complete the following fields.

Option	Description
CloudStreams Project	Click **Browse** and select the CloudStreams project in which to create the policy. Select a CloudStreams Governance project if the policy is for a virtual service or a Connector Virtual Service, or select a CloudStreams Provider project if the policy is for a custom cloud connector.
Name	A name for the new policy. A policy name can contain any character, including spaces. **Note:** If you want to change the policy name after it has been created, right-click the service name in the **Policies** folder and select **Rename**.
Service Type	Specify whether the policy will be applied to a virtual service or to a Connector Virtual Service.
Description	Optional. A description for the new policy. This description appears when a user displays a list of policies in the user interface.

4. Click **Next**.

5. Click the button next to the **Criteria** field to apply the policy to services that meet certain criteria. In the Criteria dialog box that appears, specify the criteria as follows:

   a. In the **Criteria** field, specify a value (**SOAP** or **REST**) for the **Service Type** attribute. That is, specify whether the policy should apply to SOAP services or REST services.

   b. Click the plus button next to the **Criteria** field to add a new row, and choose another criteria attribute (**Name** or **Description**) and choose an operator and value for it. For example, specify whether the policy should apply to all SOAP services that are equal to, or start with, or contain the prefix `Abc_`.

   c. Repeat step 2 if desired to specify additional criteria.

   d. If you specify multiple criteria, use the **Condition** field to connect the criteria by the AND or OR operator. The default operator is AND.

   Notes:

   - If you specify no criteria, the policy will apply to all virtual services or connector virtual services (based on your selection in the wizard).
   - Caution: CloudStreams checks for policy conflicts when you deploy a virtual service. If the service has only one policy applied to it (that is, the policy you are applying here), that policy will be deployed to CloudStreams, and CloudStreams executes the policy's actions in the order in which they are specified in the policy. However, if the service has multiple

policies applied to it, a policy conflict might occur. A policy conflict might have unintended consequences. CloudStreams will warn you of policy conflicts. For more information, see "What Happens When You Deploy a Service?" on page 194.

6. In the policy editor, click **Add Action** and select the first action to include in the policy.

   Some actions have input parameters, which are displayed in the **Action Values** section of the editor. For example, the Identify Consumer action has the parameter `Anonymous Usage Allowed`.

7. Specify values for the input parameters (if any) displayed in the **Action Values** section.

8. Right-click the action name in the **Applied Actions** list. Some actions have additional parameters you can select. For detailed descriptions of all policy action parameters, see "The Policy Action Reference" on page 168.

9. Select another action to include in the policy, if desired.

   Note:
   Make sure to select the actions in the order in which you want them to run when the policy is enforced. You cannot change the order of the actions after you add them; you can only delete them and then add a new list of actions.

10. Activate the policy when you are ready to put it into effect by right-clicking the policy name in the **Policies** folder and clicking **Active**. You will not be allowed to activate the policy unless all of its action parameters have been set. If the activation is successful, the value of the **Status** field in the Properties view will change to **Active**.

    When you deploy the virtual services or Connector Virtual Services to which the policy is applied, the policy will be deployed as well.

Modifying Policies

To modify a policy

1. Open Software AG Designer and display the CloudStreams Development perspective.

2. In the CloudStreams Governance view, click the policy name in the **Policies** folder.

3. You can modify:

   a. The policy name: Right-click the policy name in the **Policies** folder and click **Rename**.

   b. The policy description in the Properties view.

   c. The policy's criteria (to change the services to which the policy applies): Click the button next to the **Criteria** field in the Properties view (or right-click the policy name in the **Policies**

folder and click **Criteria**). The Criteria dialog box is described above, in "Creating Policies" on page 162.

d. The policy's actions and their parameters in the **Applied Actions** and **Action Values** sections.

e. To delete the policy, right-click the policy name in the **Policies** folder and click **Delete**.

4. Activate the changed policy when you are ready to put it into effect by right-clicking the policy name in the **Policy** folder and clicking **Active**. You will not be allowed to activate the policy unless all of its action parameters have been set. If the activation is successful, the value of the **Status** field in the Properties view will change to **Active**.

   When you deploy the virtual services or Connector Virtual Services to which the policy is applied, the policy will be deployed as well.

The Policy Actions

This section describes the set of policy actions that is installed with CloudStreams. The content is organized under the following sections:

- "Action Evaluation Order and Dependencies" on page 165: Lists the order in which CloudStreams evaluates the actions, lists action dependencies, and presents usage cases for authenticating/identifying consumers.
- "The Policy Action Reference" on page 168: An alphabetic reference of the actions and their parameters.

Action Evaluation Order and Dependencies

The following table shows:

- The order in which CloudStreams evaluates the actions to determine possible policy conflicts. CloudStreams resolves policy conflicts as described in "What Happens When You Deploy a Service?" on page 194.
- Any action dependencies (that is, whether an action must be used in conjunction with another particular action).

Evaluation Order	Action	Dependency
1	Require SSL	None
2	Require HTTP Basic Authentication	The "Identify Consumer" action (if the **Authenticate Credentials** option is selected).
3	Require WSS Username Token	None

Evaluation Order	Action	Dependency
4	Require X.509 Token	None
5	Require SAML Token	None
6	Require Signing	None
7	Require Encryption	None
8	Require Timestamps	The actions “Require SSL”, “Require Signing” *and* “Require Encryption”.
9	Identify Consumer	If “Identify Consumer”'s identifier field is set to: ■ **HTTP Token**, the action “Require HTTP Basic Authentication” may also be required. ■ **WSS Header Token**, the action “Require WSS Username Token” is also required. ■ **Consumer Certificate**, the actions “Require X.509 Token” or “Require Signing” are also required.
10	Authorize User	The “Require HTTP Basic Authentication”, “Require WSS Username Token” *or* “Require SAML Token” action.
11	Validate Schema	None
12	Log Invocation	None
13	Monitor Service Performance	None
14	Monitor Service Level Agreement	“Identify Consumer”
15	Throttling Traffic Optimization	“Identify Consumer” (if the **Limit Traffic for Applications** option is selected).

Usage Cases for Identifying/Authenticating Consumers

When deciding which type of identifier to use to identify a consumer application, consider the following points:

- Whatever identifier you choose to identify a consumer application, it must be unique to the application. Identifiers that represent user names are often not suitable because the identified users might submit requests for multiple applications.

- Identifying applications by IP address or host name is often a suitable choice, however, it does create a dependency on the network infrastructure. If a consumer application moves to a new machine, or its IP address changes, you must update the identifiers in the application asset.
- Using X.509 certificates or a custom token that is extracted from the SOAP message itself (using an XPATH expression), is often the most trouble-free way to identify a consumer application.

Following are some combinations of actions you can use to identify/authenticate consumers.

- Scenario 1: Identify consumers by IP address or host name

  The simplest way to identify consumers is to use the Identify Consumer action and select either the **IP Address** or **Host Name** parameter.

- Scenario 2: Authenticate consumers by HTTP authentication token

  Use the following actions:

  - Identify Consumer action, and select the **HTTP Authentication Token** parameter (to identify consumers using the token derived from the HTTP header).
  - Require HTTP Basic Authentication action.
  - Authorize User action (to authorize a list of users and/or groups registered in the Integration Server on which CloudStreams is running).

- Scenario 3: Authenticate consumers by WS-Security authentication token

  Use the following actions:

  - Identify Consumer action, and select the **WS-Security Authentication Token** parameter (to identify consumers using the token derived from the WSS Header).
  - Require WSS Username Token action.
  - Authorize User action (to authorize a list of users and/or groups registered in the Integration Server on which CloudStreams is running).

- Scenario 4: Authenticate consumers by WSS X.509 token

  Use the following actions:

  - Identify Consumer action, and select the **Consumer Certificate** parameter (to identify consumers using the WSS X.509 token).
  - Require X.509 Token action.
  - Require SSL action.

Multiple Security Elements in Requests/Responses

SOAP allows you to send multiple security elements in the SOAP header of a request or response. When CloudStreams receives a message with multiple security elements in the SOAP header, it will process only the first security element listed. It ignores all other security elements in the request.

In order for CloudStreams to process the security element, the virtual service must be configured with the proper policy action to handle the element (for example, “Require WSS Username Token”, “Require X.509 Token”, etc.). If the proper policy actions are *not* configured for the virtual service, CloudStreams will not process the security header (even if the `mustUnderstand` attribute of the security element is set to 1 (true)). In this case, CloudStreams will forward the message to the native service or the consumer unchanged.

If the proper policy actions are configured for the virtual service, CloudStreams processes the requests/responses as follows:

1. CloudStreams processes the first security element found in the message.
2. CloudStreams removes the security element from the message before sending it to the native service or the consumer.
3. If the security policy has been violated, CloudStreams sends a policy violation event notification (assuming that the policy is configured for policy violation event notifications).
4. Processing is complete; CloudStreams ignores all but the first security element.

The Policy Action Reference

This section describes, in alphabetic order, each policy action provided by CloudStreams:

Authorize User

Note:
Dependency requirement: A policy that includes this action must also include *one* of the following actions: Require HTTP Basic Authentication, Require WSS Username Token or Require SAML Token.

This action authorizes consumers against a list of users and/or a list of groups registered in the Integration Server on which CloudStreams is running.

Input Parameters

User	Right-click the action name and click **Add User** to authorize consumers against a list of users who are registered in the Integration Server instance on which CloudStreams is running. You can select **Add User** multiple times to add multiple users.
Group	Right-click the action name and click **Add Group** to authorize consumers against a list of groups who are registered in the Integration Server instance on which CloudStreams is running. You can select **Add Group** multiple times to add multiple groups.

Identify Consumer

Specifies the kind of consumer identifier (IP address, HTTP authorization token, etc.) that CloudStreams will use to identify consumer applications. You can select only one identifier. Alternatively, you can allow anonymous users to send requests, without restriction.

Input Parameters

Anonymous Usage Allowed	Allows all users to send requests, without restriction. ■ **True**: Allows all users to send requests. ■ **False**: Default. Allows only the users identified by your selected identifier (below) to send requests.
IP Address	Right-click the action name and click **Add IP Address** to identify consumer applications based on their originating IP addresses.
Host Name	Right-click the action name and click **Add Host Name** to identify consumer applications based on a host name.
HTTP Token	Right-click the action name and click **Add HTTP Token** if you want to use HTTP Basic authentication to verify the consumer's authentication credentials contained in the request's Authorization header. CloudStreams authorizes the credentials against the list of users registered in the Integration Server on which CloudStreams is running. This type of

consumer authentication is referred to as “preemptive authentication”. If you want to use “preemptive authentication”, you should also include the action “Require HTTP Basic Authentication” in the policy.

If you choose to omit “Require HTTP Basic Authentication”, the client will be presented with a security challenge. If the client successfully responds to the challenge, the user is authenticated. This type of consumer authentication is referred to as “non-preemptive authentication”. For more information, see “Require HTTP Basic Authentication” on page 181.

Note:
If you select the value **HTTP Token**, do not include the “Authorize Against Registered Consumers” action in the policy. This is an invalid combination.

WSS Header Token

Right-click the action name and click **Add WSS Header Token** to validate user names and passwords that are transmitted in the SOAP message header in the WSS Username Token. If you select this option, you should also include the action **Require WSS Username Token** in the policy.

XPATH Token

Right-click the action name and click **Add XPATH Token** to validate consumer applications based on an XML element (represented by an XPATH expression you specify in the **XPATH to Identify Token** field).

Consumer Certificate

Right-click the action name and click **Add Consumer Certificate** to identify consumer applications based on information in a WSS X.509 certificate. If you select this option, you should also include the **Require X.509 Token** or the **Require Signing** action in the policy.

User ID

If you are applying the Identify Consumer action to a connector virtual service, right-click the action name and click **Add User ID**. No other identifier is valid if you are applying the Identify Consumer action to a connector virtual service.

User ID identifies consumer applications based on a list of users who are registered in the Integration Server on which CloudStreams is running. (You need to apply the Identify Consumer action to a Connector Virtual Service if you apply the following actions to the Connector Virtual Service: “Monitor Service Level Agreement” or “Throttling Traffic Optimization” (if you select its **Limit Traffic for Applications** option)).

Include Timestamps

Note:
Dependency requirement: A policy that includes this action must also include *all* of the following actions: Require SSL, Require Signing, and Require Encryption.

If you include this action, CloudStreams will require that timestamps be included in the request header. This action supports WS-SecurityPolicy 1.2 and cannot be used with REST services or connector virtual services.

CloudStreams checks the timestamp value against the current time to ensure that the request is not an old message, which serves to protect your system against attempts at message tampering, such as replay attacks.

CloudStreams rejects the request if either of the following things happen:

- CloudStreams receives a timestamp that exceeds the time defined by the timestamp element.
- A timestamp element is not included in the request.

Input Parameters

None.

Log Invocation

Logs request/response payloads. You can log the payloads in the database and/or send the payloads in the form of email alerts. This action also logs other information about the request/response, including the service name, operation name, the Integration Server user, a timestamp, the response time, and more.

General Data Protection Regulation (GDPR) considerations

As part of this policy action, all request/response payloads are logged in the database or can be sent in the form of email alerts. Depending on the SaaS provider requirements, some of the logged data may be personal data or personally identifiable information. If for privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), you want to delete the personal data of a user, you can manage the personally identifiable information by purging it from the database.

The database used by CloudStreams has the following tables which contain request/response payloads or such information of interest. To purge the data along with the table names, the following field/column names are provided using which the data can be cleaned up.

CLS_TXN_EVENT

- REQUEST
- RESPONSE
- CONSUMER_IP

CLS_ERR_EVENT

- CONSUMER_IP

CLS_PV_EVENT

- CONSUMER_IP

CLS_MON_EVENT

- CONSUMER_IP

For email alerts, data is stored in the Integration Server logs. The log data may contain account specific or user specific data, which may be considered as personal data or personally identifiable information. If for privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), you want to delete the personal data of a user, you can manage the log files that contain the data from the file system. See the *webMethods Integration Server Administrator's Guide* for information on how to manage these logs, their related files, and specific identifiers to look for in the logs.

Input Parameters

Log Generation Frequency

Specifies how frequently to log the payload.

- **Always**: Log all requests and/or responses.
- **On Success**: Log only the successful requests and/or responses.
- **On Failure**: Log only the failed requests and/or responses.

Log the Following Payloads

Specifies whether to log all request payloads, all response payloads, or both.

- **Log Request**: Log all request payloads.
- **Log Response**: Log all response payloads.

Send Data To

Specify where to log the payloads.

- **Database**: Default. Logs the payloads in the CloudStreams Analytics database.

  Note:
  Ensure that you also select the **Database Publishing** option in Integration Server Administrator (go to **Solutions > wst > Administration > Database**), as described in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

- **Server Log**: Logs the payloads in the server log of the Integration Server on which CloudStreams is running.

  Also choose a value in the **Log Level** field:

  - Info: Logs error-level, warning-level, and informational-level alerts.
  - Warn: Logs error-level and warning-level alerts.
  - Error: Logs only error-level alerts.

Note:
The Integration Server Administrator's logging level for CloudStreams should match the logging level specified for this action (in the Integration Server Administrator go to **Settings > Logging > Server Logger**).

Alert Email

Right-click the action name and click **Add Alert Email** to send the payloads in an email alert to the email address you specify in the **Email ID** field. You can select **Add Alert Email** multiple times to add multiple email addresses.

Note:
Ensure that you set the email options in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Email**), as described in "Setting the E-mail Options for Logging Payloads and Sending Performance Monitoring Alerts" on page 31.

Monitor Service Performance

This action monitors a user-specified set of run-time performance conditions for a virtual service, and sends alerts to a specified destination when the conditions are violated.

For the counter-based metrics (Total Request Count, Success Count, Fault Count), CloudStreams sends an alert as soon as the performance condition is violated, without having to wait until the end of the metrics tracking interval. CloudStreams sends only one alert the first time the condition is violated during the interval. (It will send another alert the next time a condition is violated during a subsequent interval.) For information, see "The Intervals for Metric Publishing" on page 33.

For the aggregated metrics (Average Response Time, Minimum Response Time, Maximum Response Time), CloudStreams aggregates the response times at the end of the interval, and then sends an alert if the performance condition is violated.

Note:
By default, this action does not include metrics for failed invocations. To include metrics for failed invocations, set the `pg.PgMetricsFormatter.includeFaults`parameter to true in *`IntegrationServer_directory`*`\instances\`*`instance_name`*`\packages\WmCloudStreams\config\resources\wst-config.properties`. For more information, see "Advanced Settings" on page 281.

Input Parameters

Alert Interval

The time period (in minutes) in which to monitor performance before sending an alert if a condition is violated.

Alert Frequency

Specifies how frequently to issue alerts.

- **Every Time**: Issue an alert every time one of the specified conditions is violated.

- **Only Once**: Issue an alert only the first time one of the specified conditions is violated.

Alert Message

A text message to include in the alert.

Send Data To

Specify where to log the alerts.

- **Database**: Default. Logs the alerts in the CloudStreams Analytics database.

  Note:
  Ensure that you also select the **Database Publishing** option in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Database**), as described in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

- **Server Log**: Logs the alerts in the server log of the Integration Server on which CloudStreams is running.

  Also choose a value in the **Log Level** field:

  - Info: Logs error-level, warning-level, and informational-level alerts.
  - Warn: Logs error-level and warning-level alerts.
  - Error: Logs only error-level alerts.

  Note:
  The Integration Server Administrator's logging level for CloudStreams should match the logging level specified for this action (in the Integration Server Administrator go to **Settings > Logging > Server Logger**).

Metrics Collection Level

The run-time performance metrics for a virtual service (which is invoked only in the inbound run-time scenario), are collected at the service level. That is, the metrics for all invocations of a single virtual services are aggregated together during your specified metrics publishing interval and then published.

In contrast, the metrics for a connector virtual service (which is invoked only in the outbound run-time scenario) can be collected at two different levels of metric collection:

- **Cloud Connector Service**: Remember that a single connector virtual service can be used by multiple cloud connector services. Select this option if you want to collect the metrics for the connector virtual service broken down by each separate cloud connector service that uses it. For example, if a connector virtual services is used by three cloud connector services, then this option will collect the metrics of

that service separately, broken down by each of the three cloud connector services that use it.

- **Connector Virtual Service**(default): Select this option if you want to aggregate all the metrics for a single connector virtual service, even if it is used by multiple cloud connector services. For example, if a connector virtual service is used by three cloud connector services, then this option will collect the combined metrics for the connector virtual service by all three of the cloud connector services that use it.

Action Configuration

Right-click the action name and click **Add Action Configuration** to specify a condition to monitor. To do this, select a condition **Name** (the metric to monitor), an **Operator**, and a **Value** for the condition. You can select **Add Action Configuration** multiple times to add multiple conditions. Multiple conditions are connected by the AND operator.

Name: The metric to monitor, which can be:

- **Availability**: Indicates whether the service was available to the specified consumers in the current interval. A value of 100 indicates that the service was always available. If invocations fail due to policy violations, this parameter could still be as high as 100. That is, SOAP faults returned by the native provider or faults due to CloudStreams policy enforcements do not impact Availability. Only errors that CloudStreams interprets as a provider service being down will impact Availability.
- **Average Response**: The average amount of time it took the service to complete all invocations in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller.
- **Fault Count**: The number of faults returned in the current interval.
- **Maximum Response** : The maximum amount of time to respond to a request in the current interval.
- **Minimum Response**: The minimum amount of time to respond to a request in the current interval.
- **Successful Request Count**: The number of successful requests in the current interval.
- **Total Request Count**: The total number of requests (successful and unsuccessful) in the current interval.

Alert Email

Right-click the action name and click **Add Alert Email** if you want to send the monitoring alerts to an email address you specify in the **Email ID** field. You can select **Add Alert Email** multiple times to add multiple email addresses.

Note:
Ensure that you set the email options in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Email**), as described in "Setting the E-mail Options for Logging Payloads and Sending Performance Monitoring Alerts" on page 31.

Monitor Service Level Agreement (SLA)

Note:
Dependency requirement: A policy that includes this action must also include the Identify Consumer action.

This action is similar to the Monitor Service Performance action. Both actions can monitor the same set of run-time performance conditions for a virtual service, and then send alerts when the performance conditions are violated. This action is different because it enables you to monitor run-time performance for *one or more specified consumers*.

You can configure this action to define a *Service Level Agreement* (SLA), which is set of conditions that defines the level of performance that a consumer should expect from a service. You can use this action to identify whether a service's threshold rules are met or exceeded. For example, you might define an agreement with a particular consumer that sends an alert to the consumer if responses are not sent within a certain maximum response time. You can configure SLAs for each virtual service/consumer application combination.

For the counter-based metrics (Total Request Count, Success Count, Fault Count), CloudStreams sends an alert as soon as the performance condition is violated, without having to wait until the end of the metrics tracking interval. You can choose whether to send an alert only once during the interval, or every time the violation occurs during the interval. (CloudStreams will send another alert the next time a condition is violated during a subsequent interval.) For information about intervals, see "The Intervals for Metric Publishing" on page 33.

For the aggregated metrics (Average Response Time, Minimum Response Time, Maximum Response Time), CloudStreams aggregates the response times at the end of the interval, and then sends an alert if the performance condition is violated.

Note:
By default, this action does not include metrics for failed invocations. You can include metrics for failed invocations by setting the `pg.PgMetricsFormatter.includeFaults`parameter to true in *`IntegrationServer_directory`*`\instances\`*`instance_name`*`\packages\WmCloudStreams\config\resources\wst-config.properties`. For more information, see "Advanced Settings" on page 281.

Input Parameters

Alert Interval	The time period (in minutes) in which to monitor performance before sending an alert if a condition is violated.
Alert Frequency	Specifies how frequently to issue alerts.

- **Every Time**: Issue an alert every time one of the specified conditions is violated.
- **Only Once**: Issue an alert only the first time one of the specified conditions is violated.

Alert Message

A text message to include in the alert.

Send Data To

Specify where to log the alerts.

- **Database**: Default. Logs the alerts in the CloudStreams Analytics database.

  Note:
  Ensure that you select the **Database Publishing** option in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Database**), as described in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

- **Server Log**: Logs the alerts in the server log of the Integration Server on which CloudStreams is running.

  Also choose a value in the **Log Level** field:

  - Info: Logs error-level, warning-level, and informational-level alerts.
  - Warn: Logs error-level and warning-level alerts.
  - Error: Logs only error-level alerts.

  Note:
  The Integration Server Administrator's logging level for CloudStreams should match the logging level specified for this action (in the Integration Server Administrator go to **Settings > Logging > Server Logger**).

Metrics Collection Level

The run-time performance metrics for a virtual service (which is invoked only in the inbound run-time scenario), are collected at the service level. That is, the metrics for all invocations of a single virtual services are aggregated together during your specified metrics publishing interval and then published.

In contrast, the metrics for a connector virtual service (which is invoked only in the outbound run-time scenario) can be collected at two different levels of metric collection:

- **Cloud Connector Service**: Remember that a single connector virtual service can be used by multiple cloud connector services. Select this option if you want to collect the metrics for the connector virtual

service broken down by each separate cloud connector service that uses it. For example, if a connector virtual services is used by three cloud connector services, then this option will collect the metrics of that service separately, broken down by each of the three cloud connector services that use it.

- **Connector Virtual Service**(default): Select this option if you want to aggregate all the metrics for a single connector virtual service, even if it is used by multiple cloud connector services. For example, if a connector virtual service is used by three cloud connector services, then this option will collect the combined metrics for the connector virtual service by all three of the cloud connector services that use it.

Action Configuration

Right-click the action name and click **Add Action Configuration** to specify a condition to monitor. To do this, select a condition **Name** (the metric to monitor), an **Operator**, and a **Value** for the condition. You can select **Add Action Configuration** multiple times to add multiple conditions. Multiple conditions are connected by the AND operator.

Name: The metric to monitor, which can be:

- **Availability**: Indicates whether the service was available to the specified consumers in the current interval. A value of 100 indicates that the service was always available. If invocations fail due to policy violations, this parameter could still be as high as 100. That is, SOAP faults returned by the native provider or faults due to CloudStreams policy enforcements do not impact Availability. Only errors that CloudStreams interprets as a provider service being down will impact Availability.
- **Average Response**: The average amount of time it took the service to complete all invocations in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller.
- **Fault Count**: The number of faults returned during the current interval.
- **Maximum Response** : The maximum amount of time to respond to a request in the current interval.
- **Minimum Response**: The minimum amount of time to respond to a request in the current interval.
- **Successful Request Count**: The number of successful requests in the current interval.
- **Total Request Count**: The total number of requests (successful and unsuccessful) in the current interval.

Alert for Applications	Right-click the action name and click **Add Alert for Applications** to specify the consumer application to which this Service Level Agreement will apply. You can select **Add Alert for Applications** multiple times to add multiple consumer applications.
Add Alert Email	Right-click the action name and click **Add Alert Email** if you want to send the monitoring alerts to an email address you specify in the **Email ID** field. You can select **Add Alert Email** multiple times to add multiple email addresses.

Note:
Ensure that you set the email options in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Email**), as described in "Setting the E-mail Options for Logging Payloads and Sending Performance Monitoring Alerts" on page 31.

Require Encryption

This action requires that an XML element (which is represented by an XPath expression) be encrypted. This action supports WS-SecurityPolicy 1.2 and cannot be used with REST services or connector virtual services.

Prerequisites:

1. Configure Integration Server: Set up keystores and truststores in Integration Server (see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*).
2. Configure CloudStreams: In the Integration Server Administrator, navigate to **Solutions > CloudStreams > Administration > General** and complete the **IS Keystore Name**, **IS Truststore Name** and **Alias (signing)** fields, as described in "Setting the General Options" on page 26).

When this policy action is set for the virtual service, CloudStreams provides decryption of incoming requests and encryption of outgoing responses. CloudStreams can encrypt and decrypt only individual elements in the SOAP message body that are defined by the XPath expressions configured for the policy action. CloudStreams requires that requests contain the encrypted elements that match those in the XPath expression. You must encrypt the entire element, not just the data between the element tags. CloudStreams rejects requests if the element name is not encrypted.

Note:
Do not encrypt the entire SOAP body because a SOAP request without an element will appear to CloudStreams to be malformed.

CloudStreams attempts to encrypt the response elements that match the XPath expressions with those defined for the policy. If the response does not have any elements that match the XPath expression, CloudStreams will not encrypt the response before sending. If the XPath expression resolves a portion of the response message, but CloudStreams cannot locate a certificate to encrypt

the response, then CloudStreams sends a SOAP fault exception to the consumer and a Policy Violation event notification to CloudStreams.

How CloudStreams Encrypts Responses:

The Require Encryption action encrypts the response back to the client by dynamically setting a public key alias at run time. CloudStreams determines the public key alias as follows:

1. If CloudStreams can access the X.509 certificate of the client (based on the incoming request signature), it will use `useReqSigCert`as the public key alias.

   OR

2. If the Identify Consumer action is present in the policy (and it successfully identifies a consumer application), then CloudStreams will look for a public key alias with that consumer name in the `IS Keystore Name` property. The `IS Keystore Name` property is specified in the Integration Server Administrator, under **Solutions > CloudStreams > Administration > General**. This property should be set to an Integration Server keystore that CloudStreams will use.

   For an Identify Consumer action that allows for anonymous usage, CloudStreams does not require a consumer name in order to send encrypted responses. In this case, CloudStreams can use one of the following to encrypt the response in the following order, depending on what is present in the security element:

   - A signing certificate.
   - Consumer name.
   - WSS username, SAML token or X.509 certificate.
   - HTTP authorized user.

   OR

3. If CloudStreams can determine the current IS user from the request (that is, if an Integration Server WS-Stack determined that Subject is present), then the first principal in that subject is used.

   OR

4. If the above steps all fail, then CloudStreams will use either the WS-Security username token or the HTTP Basic-Auth user name value. There should be a public key entry with the same name as the identified username.

Input Parameters

Element Required to be Encrypted	An XPath expression that represents the XML element that is required to be encrypted.
Namespace Prefix	Optional. Right-click the action name and click **Add Namespace Prefix** if you want to specify the namespace prefix of the element required to be encrypted. Enter the namespace prefix in the following format:

`xmlns:prefix-name`

For example:

`xmlns:soapenv`

For more information, see the XML Namespaces specifications at http://www.w3.org/TR/REC-xml-names/#ns-decl.

See below for an example XPath element generated in the policy.

Example Generated XPath Element

The generated XPath element in the policy should look similar to this:

```
<sp:SignedElements xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">

  <sp:XPath
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">//soapenv:Body</sp:XPath>
</sp:SignedElements>
```

Require HTTP Basic Authentication

This action uses HTTP Basic authentication to verify the consumer's authentication credentials contained in the request's Authorization header. CloudStreams authorizes the credentials against the list of users registered in the Integration Server on which CloudStreams is running. This action supports WS-SecurityPolicy 1.2.

If the user/password value in the Authorization header cannot be authenticated as a valid Integration Server user (or if the Authorization header is not present in the request), a 500 SOAP fault is returned, and the client is presented with a security challenge. If the client successfully responds to the challenge, the user is authenticated. If the client does not successfully respond to the challenge, a 401 "WWW-Authenticate: Basic" response is returned, and the invocation is not routed to the policy engine. As a result, no events are recorded for that invocation, and its key performance indicator (KPI) data are not included in the performance metrics.

If you choose to omit the Require HTTP Basic Authentication action (regardless of whether an Authorization header is present in the request or not), then:

- CloudStreams forwards the request to the native service, without attempting to authenticate the request.
- The native service returns a 401 "WWW-Authenticate: Basic" response, which CloudStreams will forward to the client; the client is presented with a security challenge. If the client successfully responds to the challenge, the user is authenticated.

In the case where a consumer is sending a request with both transport credentials (HTTP Basic authentication) and message credentials (WSS Username or X.509 token), the message credentials take precedence over the transport credentials when Integration Server is determining which credentials it should use for the session. For more information, see "Require WSS Username Token" on page 185 and "Require X.509 Token" on page 186. In addition, you must ensure that the consumer that connects to the virtual service has an Integration Server user account.

Input Parameters

Authenticate Credentials	Authorizes consumers against the list of users registered in the Integration Server on which CloudStreams is running. If you select this option, you must also include the Identify Consumer action in the policy.

Require SAML Token

Requires that a WSS Security Assertion Markup Language (SAML) assertion token be present in the SOAP message header to validate service consumers. CloudStreams supports SAML 1.1 and 2.0 tokens. This action supports WS-SecurityPolicy 1.2 and cannot be used with REST services or connector virtual services.

Note:
When a Require SAML Token action is generated, CloudStreams also implicitly selects the `timestamp` and `signing` assertions. You should not add the Include Timestamps and Require Signing policy actions to a virtual service if the Require SAML Token action is already applied.

Input Parameters

SAML Version	**String.** Specifies the version of the WSS SAML Token to use (1.1 or 2.0).

Run-Time Behavior

When a service consumer sends a request that includes a SAML token to a virtual service, CloudStreams validates the SAML token to ensure it is valid. If the token is valid, Integration Server uses its included JAAS login modules to process the SAML assertion and map the client public key in the assertion to a valid Integration Server user.

If the service consumer invokes the virtual service without a SAML assertion in the request, then CloudStreams sends the following SOAP fault to the service consumer to indicate that the request does not match the security policy being enforced: `SAML Token missing in request`.

Prerequisites

In order to use a SAML token, CloudStreams requires that you:

- Determine which Security Token Services (STS) to trust. The STS generates the SAML tokens that clients will submit. The client can use any STS provider that generates SAML 1.1 or 2.0 tokens. The generated SAML token must:
  - Contain the certificate of the user/client (service consumer) in the assertion if Integration Server is to use Holder-of-Key (HOK) type SAML assertions.
  - Be signed by the STS.

- Identify the trusted STS to Integration Server. For more information, see "Setting the STS Options" on page 35.
- Provide a truststore alias that points to a truststore containing the issuer's certificate. For information about providing a truststore alias, see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*.
- If Integration Server is to process Holder-of-Key (HOK) type SAML assertions, which contain the client's public key, you must map the client's public key to an Integration Server user. For more information about configuring and mapping client certificates, see the section *Authenticating Clients* in the document *webMethods Integration Server Administrator's Guide*.

To Identify a Trusted STS to Integration Server

To identify a trusted STS to Integration Server

1. Open the Integration Server Administrator if it is not already open.
2. In the Navigation panel, select **Security > SAML**.
3. Click **Add SAML Token Issuer**, set the following parameters and click **Save Changes**:

Option	Description
Issuer Name	Name of a SAML token issuer from which Integration Server should accept and process SAML assertions. Integration Server will reject SAML assertions from issuers not configured on this screen and will log a message to the Server log similar to the following: `2010-06-09 23:35:38 EDT [ISS.0012.0025E` This value must match the value of the Issuer field in the SAML assertion.
Truststore Alias	Text identifier for the truststore, which contains the public keys of the SAML token issuer.
Certificate Alias	Text identifier for the certificate associated with the truststore alias.
Clock Skew	Clock difference between your Integration Server and the SAML token issuer.

Require Signing

Note:
Dependency requirement: A policy that includes this action must also include the Identify Consumer action.

This action requires that a request's XML element (represented by an XPath expression) be signed. This action supports WS-SecurityPolicy 1.2 and cannot be used with REST services or connector virtual services.

Prerequisites:

1. Configure Integration Server: Set up keystores and truststores in Integration Server (see the section *Securing Communications with the Server* in the document *webMethods Integration Server Administrator's Guide*).

2. Configure CloudStreams: In the Integration Server Administrator, navigate to **Solutions > CloudStreams > Administration > General** and complete the **IS Keystore Name**, **IS Truststore Name** and **Alias (signing)** fields, as described in "Setting the General Options" on page 26). CloudStreams uses the signing alias specified in the **Alias (signing)** field to sign the response.

When this policy action is set for the virtual service, CloudStreams validates that the requests are properly signed, and provides signing for responses. CloudStreams provides support both for signing an entire SOAP message body or individual elements of the SOAP message body.

CloudStreams uses a digital signature element in the security header to verify that all elements matching the XPath expression were signed. If the request contains elements that were not signed or no signature is present, then CloudStreams rejects the request.

Note:
You must map the public certificate of the key used to the sign the request to an Integration Server user. If the certificate is not mapped, CloudStreams returns a SOAP fault to the caller.

Input Parameters

Element Required to be Signed	An XPath expression that represents the XML element that is required to be signed.

Namespace Prefix	Optional. Right-click the action name and click **Add Namespace Prefix** if you want to specify the namespace prefix of the element required to be signed. Enter the namespace prefix in the following format:
	`xmlns:prefix-name`
	For example:
	`xmlns:soapenv`
	For more information, see the XML Namespaces specifications at http://www.w3.org/TR/REC-xml-names/#ns-decl.
	See below for an example XPath element generated in the policy.

Example Generated XPath Element

The generated XPath element in the policy should look similar to this:

```
<sp:SignedElements xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
             <sp:XPath
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">//soapenv:Body</sp:XPath>

          </sp:SignedElements>
```

Require SSL

This action ensures that requests are sent to the server using the HTTPS protocol (SSL). This action supports WS-SecurityPolicy 1.2 but can be used with both SOAP and REST services.

In addition, setting the **Client Certificate Required** parameter to True allows CloudStreams to verify the client sending the request.

Ensure that you specify an HTTPS port, as described in "Setting the General Options" on page 26.

Input Parameters

Client Certificate Required	Specifies whether client certificates are required for the purposes of: ■ Verifying the signature of signed SOAP requests or decrypting encrypted SOAP requests. ■ Signing SOAP responses or encrypting SOAP responses. Values: ■ **True**: Requires client certificates. If a valid client certificate is not presented, CloudStreams rejects the message. Ensure that the Integration Server HTTPS port is configured to request or require a client certificate. ■ **False**: Default. Does not require client certificates.

Require WSS Username Token

Note:
Dependency requirement: A policy that includes this action must also include the Identify Consumer action.

Requires that a WSS username token and password be present in the SOAP message header to validate service consumers. This action supports WS-SecurityPolicy 1.2 and cannot be used with REST services or connector virtual services.

CloudStreams rejects requests that do not include the username token and password of an Integration Server user. CloudStreams only supports clear text passwords with this kind of authentication.

In the case where a consumer is sending a request with both transport credentials (HTTP basic authentication) and message credentials (WSS username or X.509 token), the message credentials

take precedence over the transport credentials when Integration Server is determining which credentials it should use for the session. For more information, see "Require HTTP Basic Authentication" on page 181.

Input Parameters

None.

Require X.509 Token

Note:
Dependency requirement: A policy that includes this action must also include the Identify Consumer action.

Requires that a WSS X.509 token be present in the SOAP message header to validate service consumers. This action supports WS-SecurityPolicy 1.2 and cannot be used with REST services or connector virtual services.

In the case where a consumer is sending a request with both transport credentials (HTTP basic authentication) and message credentials (X.509 token or WSS username), the message credentials take precedence over the transport credentials when Integration Server is determining which credentials it should use for the session. For more information, see "Require HTTP Basic Authentication" on page 181. In addition, you must ensure that the service consumer that connects to the virtual service has an Integration Server user account.

Input Parameters

None.

Throttling Traffic Optimization

Note:
Dependency requirement: A policy that includes this action must also include the Identify Consumer action if the **Limit Traffic for Applications** option is selected.

This action limits the number of service invocations allowed during a specified time interval, and sends alerts to a specified destination when the performance conditions are violated.

Reasons for limiting the service invocation traffic include:

- To avoid overloading the back-end services and their infrastructure.
- To limit specific consumers in terms of resource usage (that is, you can use the Monitor Service Level Agreement action in addition to Throttling Traffic Optimization).
- To shield vulnerable servers, services, and even specific operations.
- For service consumption metering (for example, billable pay-per-use services).

Input Parameters

Soft Limit	**Number.** Optional. Specifies the maximum number of invocations allowed per **Interval** before issuing an alert. Reaching the soft limit will not affect further processing of requests (until the **Hard Limit** is reached). The limit is reached when the total number of invocations coming from all the consumer applications (specified in the **Limit Traffic for Applications** field) reaches the limit.
Hard Limit	**Number.** Required. Specifies the maximum number of invocations allowed per **Interval** before stopping the processing of further requests and issuing an alert. Typically, this limit should be higher than the soft limit. The limit is reached when the total number of invocations coming from all the consumer applications (specified in the **Limit Traffic for Applications** field) reaches the limit.
Alert Interval	**Number.** Specifies the amount of time for the soft limit and hard limit to be reached.
Alert Frequency	**Number.** Specifies how frequently to issue alerts. ■ **Every Time**: Issue an alert every time one of the specified conditions is violated. ■ **Only Once**: Issue an alert only the first time one of the specified conditions is violated.
Alert Message for Soft Limit	Optional. Specify a text message to include in the soft limit alert.
Alert Message for Hard Limit	Optional. Specify a text message to include in the hard limit alert.
Send Data To	Specify where to log the alerts. ■ **Database**: Default. Logs the alerts in the CloudStreams Analytics database. **Note:** Ensure that you also select the **Database Publishing** option in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Database**), as described in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32. ■ **Server Log**: Logs the alerts in the server log of the Integration Server on which CloudStreams is running.

Also choose a value in the **Log Level** field:

- Info: Logs error-level, warning-level, and informational-level alerts.
- Warn: Logs error-level and warning-level alerts.
- Error: Logs only error-level alerts.

> **Note:**
> The Integration Server Administrator's logging level for CloudStreams should match the logging level specified for this action (in the Integration Server Administrator go to **Settings > Logging > Server Logger**).

Metrics Collection Level

The run-time performance metrics for a virtual service (which is invoked only in the inbound run-time scenario), are collected at the service level. That is, the metrics for all invocations of a single virtual services are aggregated together during your specified metrics publishing interval and then published.

In contrast, the metrics for a connector virtual service (which is invoked only in the outbound run-time scenario) can be collected at two different levels of metric collection:

- **Cloud Connector Service**: Remember that a single connector virtual service can be used by multiple cloud connector services. Select this option if you want to collect the metrics for the connector virtual service broken down by each separate cloud connector service that uses it. For example, if a connector virtual services is used by three cloud connector services, then this option will collect the metrics of that service separately, broken down by each of the three cloud connector services that use it.
- **Connector Virtual Service**(default): Select this option if you want to aggregate all the metrics for a single connector virtual service, even if it is used by multiple cloud connector services. For example, if a connector virtual service is used by three cloud connector services, then this option will collect the combined metrics for the connector virtual service by all three of the cloud connector services that use it.

Limit Traffic for Application

Right-click the action name and click **Add Limit Traffic for Application** if you want to specify the consumer application that this action applies to. You can select **Add Limit Traffic for Application** multiple times to add multiple consumer applications.

Alert Email

Right-click the action name and click **Add Alert Email** if you want to send the monitoring alerts to an email address you specify in the **Email ID** field. You can select **Add Alert Email** multiple times to add multiple email addresses.

Note:
Ensure that you also set the email options in Integration Server Administrator (go to **Solutions > CloudStreams > Administration > Email**), as described in "Setting the E-mail Options for Logging Payloads and Sending Performance Monitoring Alerts" on page 31.

Validate Schema

This action validates all XML request and/or response messages against an XML schema referenced in the WSDL. This action cannot be used for connector virtual services.

CloudStreams can enforce this policy action for messages sent between services. When this policy is set for a virtual service, CloudStreams validates XML request messages, response messages, or both, against the XML schema referenced in the WSDL. Ensure that you provide a schema.

Input Parameters

Validate SOAP Message(s)	Validates the request and/or response messages.

Note:
Be aware that CloudStreams does not remove `wsu:Id` attributes that may have been added to a request by a consumer as a result of security operations against request elements (signatures and encryptions). In this case, to avoid schema validation failures you would have to add a Transform sub-step to the virtual service's In Sequence step so that the requests are passed to an XSLT transformation file that removes the `wsu:Ids`.

5 Deploying Virtual Services and Connector Virtual Services

Overview of Deployment

You can deploy virtual services and Connector Virtual Services in two ways:

- Deploy all Virtual Services, and any custom Connector Virtual Services, that are contained in a particular CloudStreams Governance project.

  OR

- Deploy a single Virtual Service or custom Connector Virtual Service. Generally, this is useful for testing purposes.

Note:
CloudStreams does not support sharing of Connector Virtual Services, Virtual Services, and Policies across nodes in a clustered setup. These artifacts should be manually deployed to a clustered node as needed.

Note:
CloudStreams automatically deploys the *default* Connector Virtual Services, `WmCloudStreams.SoapVS` and `WmCloudStreams.RestVS`; there is no need for you to deploy them.

When you execute the deployment operation, CloudStreams will immediately deploy the service(s) and the following items to the CloudStreams server target(s) that you specify:

- The policies of each service, as well as any other artifacts that you associated with the services when you created them.
- The VSD (virtual service definition) of each virtual service or Connector Virtual Service.

  When you deploy a Virtual Service or a Connector Virtual Service, CloudStreams generates an XML document called a *virtual service definition (VSD)*. The VSD defines the virtual service or Connector Virtual Service for CloudStreams, and contains all the resources required to deploy the service to a CloudStreams server, including the policy that applies to the service. You cannot edit the VSD, but you can view it in the **Advanced** page in the Properties view of each service.

Before You Deploy Virtual Services or Connector Virtual Services

You should:

- Ensure that all policies of the services are Active; the value of the **Status** field in the Properties view of each policy should be **Active**. If it is not, right-click the policy name in the CloudStreams Governance view and click **Active**. You will not be allowed to activate a policy unless all of its action parameters have been set.
- Ensure that at least one CloudStreams server target has already have been defined, as described in "Defining and Managing a CloudStreams Server Target" on page 59.
- Ensure that each target's specified deployment URL is active and the user credentials of Integration Server are valid.

- Ensure that you specify an HTTP or HTTPS port, as described in "Setting the General Options" on page 26.
- Test the server connection by going to the Software AG Designer menu, selecting **Window > Preferences > Software AG > CloudStreams Servers**, and clicking the **Test** button. If the connection is not active and valid, activate the deployment endpoint and modify the user credentials as required.

Deploying All Services in a CloudStreams Governance Project

When you deploy a CloudStreams Governance project, all virtual services in the project are deployed at once. If any custom Connector Virtual Services are defined in the project, they are deployed too.

When you deploy a project, Integration Server automatically creates a package to support the project. The package name is the same as your project name. This package includes startup/shutdown services to manage the registration of the virtual service definitions (VSDs) when Integration Server restarts.

To deploy all services within a project

1. Open Software AG Designer and display the CloudStreams Development perspective by clicking **Window > Open Perspective > Other > CloudStreams Development**.
2. In the CloudStreams Governance view, right-click the CloudStreams Governance project you want to deploy, and click **Deploy**.
3. In the Deploy dialog box, choose one or more CloudStreams server targets to which to deploy the services and click **OK**.

   Note:
   If only one CloudStreams server target is available to select, this dialog box will not appear.

   The services are immediately deployed to the following location:

   IntegrationServer_directory\instances*instance_name*\packages\<*PackageName*><*ProjectName*>\config\proxies*VirtualService*

   If CloudStreams encounters a problem deploying or redeploying a service, it displays a failure message in the deployment log at the bottom of the page. In this case, the CloudStreams administrator should take corrective action and redeploy the service. For more information, see "What Happens When You Deploy a Service?" on page 194.

   If one of the virtual services fails deployment, you do not need to redeploy the entire project; just correct the problem with the failed virtual service and then redeploy it separately.

Deploying a Single Service

To deploy a single virtual service or Connector Virtual Service

1. Open Software AG Designer and display the CloudStreams Development perspective.
2. Right-click the service name and click **Deploy** in the context menu.
3. In the Deploy dialog box, choose one or more CloudStreams server targets to which to deploy the service and click **OK**.

   Note:
   If only one CloudStreams server target is available to select, this dialog box will not appear.

   The service is immediately deployed to the following location:

   IntegrationServer_directory\instances*instance_name*\packages\<*PackageName*><*ProjectName*>\config\proxies*VirtualService*

   If CloudStreams encounters a problem deploying or redeploying a service, it displays a failure message in the deployment log at the bottom of the page. In this case, the CloudStreams administrator should take corrective action and redeploy the service. For more information, see "What Happens When You Deploy a Service?" on page 194.

Undeploying Services

To undeploy a virtual service or Connector Virtual Service

1. Open Software AG Designer and display the CloudStreams Development perspective.
2. In the CloudStreams Governance view, expand the CloudStreams Governance project that contains the services you want to undeploy.
3. Right-click either the project name (to undeploy all services) or the service name and click **Undeploy** in the context menu.

   CloudStreams immediately undeploys the service(s).

What Happens When You Deploy a Service?

When you deploy virtual services or Connector Virtual Services to CloudStreams targets, keep the following points in mind.

Policy Validation

When you deploy a service, CloudStreams automatically validates the service's policy (or policies) to ensure that all action dependencies are properly met. CloudStreams will warn you of any violation, and you will need to correct the violations before deploying the service. For more information about dependencies, see "The Policy Actions" on page 165.

Policy Conflict Resolution Rules

When the combined list of actions is evaluated, policy conflicts can arise due to multiple authentication or authorization actions. Conflicts between these actions are resolved as follows.

Action Type	Resolution
Authentication actions: ■ Require HTTP Basic Authentication ■ Require WSS Username Token ■ Require X.509 Token ■ Require SAML Token	One occurrence of each authentication action that appears in the list is chosen.
Authorization actions: ■ Authorize User	All occurrences of authorization actions are chosen.

The Policy Conflict Resolution rules evaluate the actions from all policies in the following order. The rules choose the actions for the effective policy as follows:

Order	Action	Result Chosen for Effective Policy
1	Require SSL	One occurrence of the action is chosen if one of the actions has its `Client Certificate Required` parameter set to Yes.
2	Require HTTP Basic Authentication	One occurrence of the action is chosen.
3	Require WSS Username Token	One occurrence of the action is chosen.
4	Require WSS X.509 Token	One occurrence of the action is chosen.
5	Require WSS SAML Token	One occurrence of the action is chosen.
6	Require Signing	One occurrence of the action is chosen.
7	Require Encryption	One occurrence of the action is chosen.
8	Require Timestamps	One occurrence of the action is chosen.

Order	Action	Result Chosen for Effective Policy
9	Identify Consumer	One occurrence of the action is chosen. CloudStreams evaluates the consumer identifiers in all the actions and selects the action with the highest-priority consumer identifier, as follows: 1.) IP address 2.) Host name 3.) HTTP Authentication token 4.) WS-Security token 5.) Custom identification token 6.) Consumer certificate 7.) User ID.
10	Authorize User	All occurrences of the action are chosen.
11	Validate Schema	If at least one occurrence of the action is configured to validate requests, and at least one occurrence of the action is configured to validate responses, then an action is constructed to validate both requests and responses.
12	Log Invocations	All occurrences of the action are chosen.
13	Monitor Service Performance	All occurrences of the action are chosen.
14	Monitor Service Level Agreement	All occurrences of the action are chosen.
15	Throttling Traffic Optimization	All occurrences of the action are chosen.

What if You Need to Modify Deployed Services?

If you need to modify a virtual service that is already deployed, you must redeploy it after you modify it. CloudStreams does not monitor for updates to deployed assets. If you make changes to a virtual service's processing steps, for example, you must manually redeploy the virtual service to put those changes into effect.

6 Cloud Connections, Services, and Connector Listeners

CloudStreams Provider

A provider is a logical grouping of connectors for a SaaS vendor. A provider is installed on Integration Server as an Integration Server package.

CloudStreams Connector

A connector represents an integration interface to the provider using a specific communication channel. Examples of connectors are Salesforce Partner SOAP API and Bulk REST API.

The type of a CloudStreams connector depends on the type of SaaS provider with which you communicate. The two types of connectors are as follows:

- If you communicate with a SOAP-based cloud application provider, you create cloud connections using a CloudStreams SOAP connector.
- If you communicate with a REST-based provider, you create cloud connections using a CloudStreams REST connector.

Note:
Both SOAP and REST connectors support connectivity with providers that expose Streaming API connectivity.

Cloud Connection

You can create one or more connections for a connector at design time to use in integrations. A cloud connection is required for configuring and running a cloud connector service. You must create and enable a cloud connection before you can create cloud connector services. You can also test the connection before you save the configuration of a connection. By default, the Test option is disabled. You need to update the respective connector to enable the option. See the respective provider documents for connection configurations and how to configure connections for a connector. Connection configurations vary for each connector.

Cloud Connector Service

A cloud connector service defines an interaction that the connector will perform on the SaaS back end. The cloud connector service is analogous to any other Integration Server service and could be used in a flow service. You call a cloud connector service within a flow service and you can audit them from the audit system of Integration Server.

A cloud connector service has input and output signatures. An input signature describes the data that the service expects to find in the service pipeline at run time. An output signature describes the data that the service expects to add to the pipeline when it has successfully executed. You can view the node signature of a cloud connector service on the Input/Output tab of the cloud connector service editor in Software AG Designer. Before configuring a cloud connector service, you must assign it a connection that you created earlier.

A cloud connector service is based on the interaction definitions defined in the connector and contains appropriate logic for executing the interaction and getting the response from the back end.

Based on the connector type, cloud connector service definitions allow you to configure the interactions. For REST, you need to define the Resource, HTTP Headers, and parameters. For SOAP, you need to define the Operation, SOAP Headers, and custom parameters, if any.

You can create a cloud connector service using a wizard in Software AG Designer. See the documentation specific to your CloudStreams provider, for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*.

Working with Connector Listeners

Most of the modern SaaS providers provide streaming APIs, which send data to clients using push technologies like Comet, HTTP streaming, and so on. Prior to the CloudStreams version 10.3 release, CloudStreams supported connectivity with streaming API providers as a native configuration, that is, details for user created streaming subscriptions were persisted within the WmCloudStreams package. From the CloudStreams v10.3 release, streaming API access is available as a *connector capability* and you can integrate with a streaming API using a CloudStreams connector.

CloudStreams connectors now support connectivity with streaming APIs and processing of streaming API events in addition to supporting connectivity and execution of SOAP and REST APIs. So in addition to connecting to SOAP and REST APIs, you can now use CloudStreams connections for connecting to streaming APIs. You can create a CloudStreams connector listener, select a subscription channel from a list of available channels for an endpoint, and configure the action(s) to be applied on the incoming events. Additionally, you can configure transport or communication related parameters for a CloudStreams connector listener as well as enable and disable a CloudStreams connector listener. A CloudStreams listener receives the streaming API events and processes the received events.

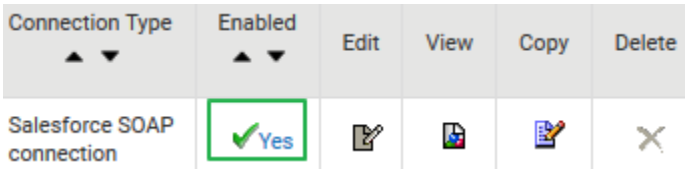
You create connector listeners using the Software AG Designer. Before you create a connector listener, ensure that the CloudStreams connector associated with your cloud application provider is installed. Also ensure that a cloud connection pool is created for that connector.

To create and configure a connector listener

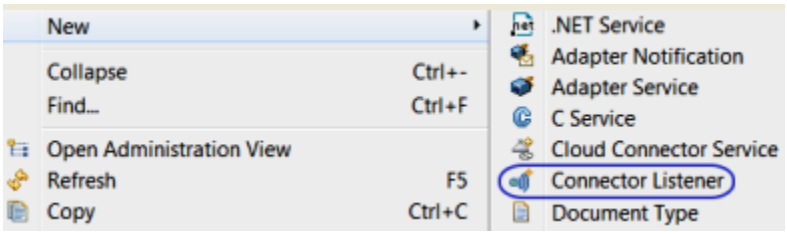
1. Install the CloudStreams provider package that has streaming API capability.

2. Go to the connection configuration page in Integration Server Administrator and create a connection for the connector. You can also use an existing connection for the selected provider and connector. The streaming functionality will leverage the existing authentication, timeouts (Connection TimeOut and Socket Read Timeout), truststore (Trust store Alias), keystore (Keystore Alias), host name verification (Hostname verifier), and proxy (Proxy Server Alias) configurations from the referenced CloudStreams connection.

3. Enable the connection in Integration Server Administrator.



4. Go to the **Service Development** perspective in Software AG Designer and create a connector listener.

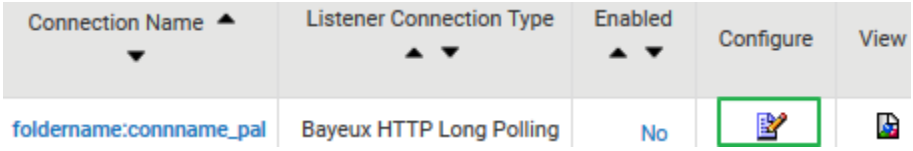


5. In the **Service Development** perspective, on the **Listener** page for the newly created connector listener, configure the subscription request, that is, select a **Subscription** from the list of available subscription channels.

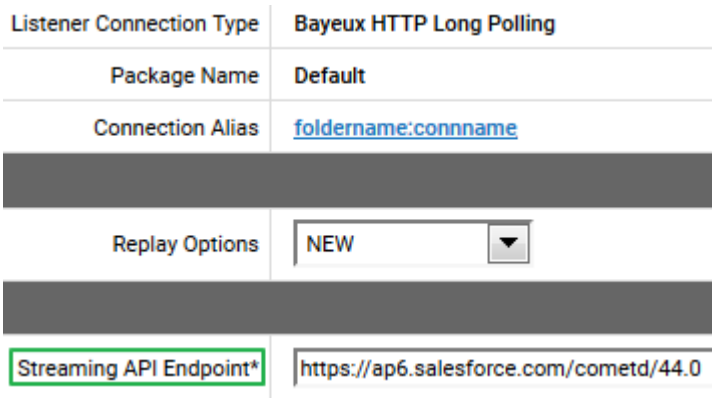
6. Configure the required headers and parameters to complete the subscription request configuration. For example, for the **Value** field in the **Parameters** page, you can add the push topic name created as a parameter. You can also select a different connection pool for the subscription request or choose a different document to represent the subscribed event.

7. Configure the processing of the subscribed events by adding an **Action** to be applied on the incoming events. You can log or invoke a service based on your configurations.

8. To manage the connector listener, go to Integration Server Administrator and in the **Listeners** page for the connector, click **Configure**.



9. Edit the listener communication or transport specific parameters for accessing the streaming API.



10. For the selected connector listeners, you can configure error recovery and callback information.

    When a Streaming Provider triggers error events such as *403::Unkown Client or 401::Authentication invalid*, the error handler service is invoked to identify if the error can be recovered. If the error is recoverable, then an appropriate action is taken to restore the listener to the function. The default callback service is currently configured to log the error and error recovery status. Thus, you can ensure that the listener is fault tolerant by enabling the error handler, and listener runtime errors are reported and appropriate action taken.

    For more information on the listener error handler functionality, see the documentation specific to your CloudStreams provider, for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*.

11. Enable the listener in Integration Server Administrator.

    Once the listener is enabled, connection with the streaming API is established, and incoming events will be processed whenever the streaming Provider triggers the events.

    For more information about CloudStreams connector listeners, see the *webMethods Service Development Help* and the documentation specific to your CloudStreams provider, for example, the *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*. For information on the deployable assets, see the *webMethods Deployer User's Guide*.

Replaying Salesforce events

CloudStreams connector provides a near real-time integration with Salesforce by leveraging their Streaming APIs based on Bayeux protocol and CometD. The underlying communication is based on long polling where the connector listener establishes a persistent connection with the Salesforce provider. This connection remains open while transmitting the events until either side closes the connection. Through this established connection, events are streamed to the listener. If the connection is lost due to a network failure or any other reason, you can retrieve the standard-volume events that are within Salesforce's 24-hour retention window.

Note:

You can retrieve Salesforce events only if you are using Salesforce v37.0 or later and CloudStreams v10.3 or later.

Each Salesforce event is assigned an opaque ID which is unique for each event. This ID is contained in the *replayId* field of the event object. The *replayId* field value which is populated by Salesforce when the event is delivered to subscribers, refers to the position of the event in the event stream. For consecutive events, *replayId* values may have a break between them, for example, the event with ID 110 may follow the event with ID 101.

Example of an event object that Salesforce sends to its subscribed clients

```
{
          "clientId":"a1ps4wpe52qytvcvbsko09tapc",
          "data":{
             "event":{
                "createdDate":"2016-03-29T19:05:28.334Z",
                "replayId":55
                },
             "payload":"This is a message."
             },
          "channel":"/u/TestStreaming"
          }
```

To replay events, Salesforce provides a way to configure the *replayId* while subscribing to a particular channel. You can replay the lost Salesforce events for a Cloudstreams connector listener by specifying the **Replay Options** using the CloudStreams Administration console in Integration Server Administrator.

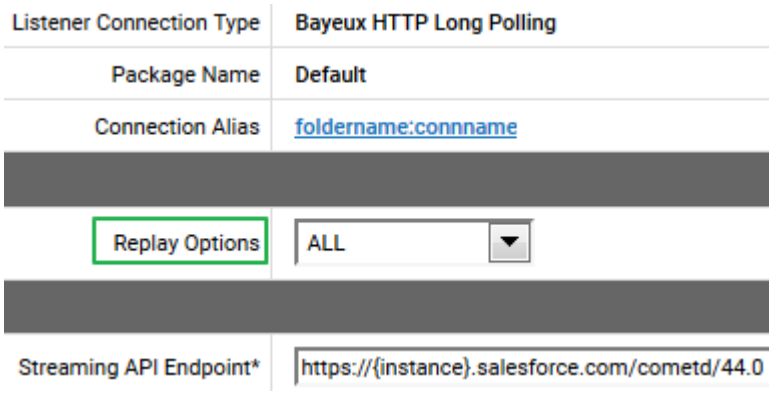
Replay Option (replayId)	Usage
NEW (-1)	Receive new events that are broadcast after the client subscribes.
ALL (-2)	Receive all events including past events that are within the retention window including new events.
LAST-RECEIVED	Receive all the events after the last received event by CloudStreams within the retention window. This configuration is helpful especially in cases such as sudden missing events due to a network failure or any other unexpected error. For example, if the connector listener subscription is active, and a temporary network disruption can cause a loss of events. However, if you have configured the connector listener with this option then the missing events are automatically replayed as soon as the connectivity is restored with a *replay id* that was last received by CloudStreams before the network failure.
CUSTOM	Receive all the events specified by a *replayId* computed by a custom service. You can store and compute the *replayId*s of the past events. This option lets you point to any *replay id* in the old event stream. You must write a custom service adhering to

Replay Option (replayId)	Usage
	`wm.cloudstreams.listener.metadata.connection.specs:retrieveReplayIDSpec` and it must map the *replayId* to `computedReplayID` key. This service must be provided in the **Retrieve Replay ID Service** field. The configured service is executed that is subscribing to Salesforce Streaming API and the value of the `computedReplayID` is used to specify the *replayId* during subscription.

Note:
Currently, the replay options are supported only for Salesforce Push Topic.

To replay Salesforce events

1. Go to Integration Server Administrator > Solutions > CloudStreams > Providers.
2. Select the Salesforce.com provider to list all the connectors.
3. Click on the connector for which you have already created a listener.
4. Go to the **Listener** page and *disable* the Salesforce connector listener for which you want to configure the replay option.
5. Click **Configure**.
6. In the **Replay Options** field, choose:
   - **ALL** - To replay all the events for the last 24 hours.
   - **NEW** - To replay only the new events from the time the connector listener is enabled.



   - **LAST-RECEIVED**- To replay all the events after the event that was last received by CloudStreams, within the retention window.

- **CUSTOM**- To replay the events after the event specified by a *replayId* that is computed by a custom service. The customized service is provided in the **Retrieve ReplayID Service** field.

7. Click **Enable**

   While enabling the listener, you can choose to provide a *replayId* in the **Specify ReplayID** field. This is an *optional* field. If you provide the *replayId* in this field, then all the events past that *replayId* will be replayed and the action cannot be undone. This input *replayId* overrides all the other configurations. If you choose to ignore this field, then the *replayId* derived from the **Replay Option** setting will take effect. Additionally, this *replayId* can only be tracked as part of auditing or logging as it is not stored within CloudStreams.

Detection and processing of duplicate Salesforce events

If you select **ALL** as the replay option, Salesforce sends all the events that are stored within the last 24-hour window. You may get events that were already received and these events are called duplicate events. CloudStreams provides a duplicate detection mechanism which is enabled by default. The duplicate detection mechanism checks if an event is a duplicate event based on the event extractor information that is defined as part of the event definition.

Duplicate event detection mechanism is offered on a best-efforts basis. CloudStreams internally uses persistent caches to perform duplicate detection. You must estimate the maximum load for the system and tune the various cache configurations accordingly. For guaranteed duplicate detection, tune the cache so that it can accommodate the highest payload effectively.

For example, if you expect a maximum of 10000 events in a day from Salesforce and select the **ALL** replay option, you must configure the *maxElementsInMemory* property of the *ListenerEventsCache* cache to be well above 10000. Set this configuration by editing the `SoftwareAG-IS-CloudStreams.xml` cache file available at `<IS-installation-dir>\IntegrationServer\instances\default\packages\WmCloudStreams\config\resources\caching`.

To know more about the configuration options for EhCache and how to size caches, see the Ehcache documentation at http://www.ehcache.org/documentation/.

If the cache is not configured specifically for this purpose, and if duplicate Salesforce events are not detected, you must implement some custom mechanism or custom handling to stop the processing of those duplicate events.

Persistent configuration of events for duplicate detection

Persistent configuration of events is not configured by default. If Integration Server is operating in a clustered mode, then persistent configuration of events will be automatically enabled. To

enable persistent configuration of events for an on-premises stand-alone Integration Server installation, which is *not* in a clustered setup, do the following:

1. Go to the `<IS-installation-dir>\IntegrationServer\instances\<instance name>\packages\WmCloudStreams\config\resources\caching` directory.

2. Open the `SoftwareAG-IS-CloudStreams.xml` file.

3. In the *ListenerEventsCache* cache, specify the following entry: `<persistence strategy="localrestartable" synchronousWrites="true"/>`

4. Delete the CloudStreams caching file `SoftwareAG-IS-CloudStreams.xml`, from the location, `<IS-installation-dir>\IntegrationServer\instances\<instance name>\config\caching` for the above changes to take effect.

5. Restart Integration Server.

Persistent configuration of last received Replay IDs

CloudStreams internally uses persistent caches to store *replayIds* for each listener. Do the following for each setup of CloudStreams to persist entries of the cache:

- CloudStreams setup in a cluster: For a cluster setup, the persistence of this cache will be handled by default.
- CloudStreams setup in a stand-alone mode: For any on-premises stand-alone Integration Server installation, which is not in a clustered setup, the administrator will have to do the following:

1. Go to the `<IS-installation-dir>\IntegrationServer\instances\<instance name>\packages\WmCloudStreams\config\resources\caching` directory.

2. Open the `SoftwareAG-IS-CloudStreams.xml` file.

3. In the `ReplayIDCache` cache, specify the following entry: `<persistence strategy="localrestartable" synchronousWrites="true"/>`

4. Delete the CloudStreams caching file `SoftwareAG-IS-CloudStreams.xml` file, from the location, `<IS-installation-dir>\IntegrationServer\instances\<instance name>\config\caching` for the above changes to take effect.

5. Restart Integration Server.

Configuring Error Recovery and Callback information for Salesforce

When a Streaming Provider triggers error events such as *403::Unkown Client or 401::Authentication invalid*, the CloudStreams server engine invokes error handler service and error handler service identifies whether the error is recoverable or not.

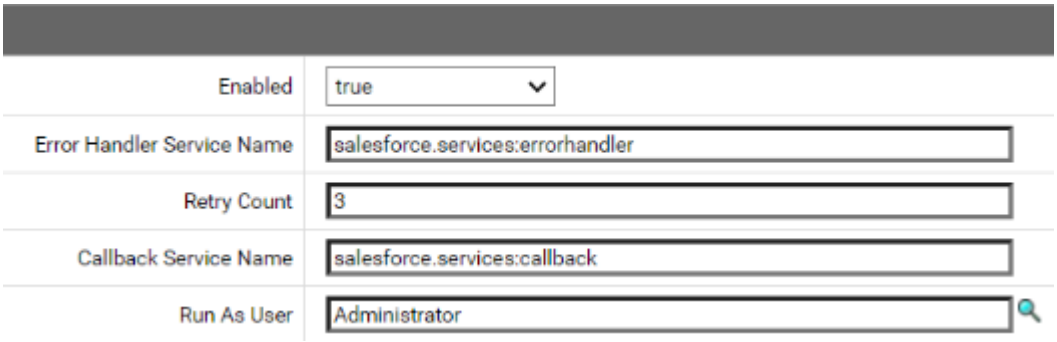
If the error is recoverable, the action is taken by the CloudStreams server engine to restore the listener to the function. Later, the callback service is invoked by the CloudStreams server engine, by providing the status of error recovery.

If the error is not recoverable, CloudStreams engine invokes the callback service, by providing the status of error recovery.

The default callback service is currently configured to log the error and error recovery status.

To configure error callback for Salesforce

1. Go to **Integration Server Administrator** > **Solutions** > **CloudStreams** > **Providers**.
2. Select the Salesforce.com provider to list all the connectors.
3. Click on the connector for which you have already created a listener.
4. Go to the **Listener** page and then go to the **Error Callback Configuration** section and configure the callback service. You can edit the callback service based on your configuration. The below image shows the configuration for Salesforce.com.



Configure the call back service:

Fields	Description
Enabled	Once you configure the listener, you can use this option's drop-down list to recover the error: ■ *true* ■ *false*

Fields	Description
	Note: By default, this field is set to *true*.
Error Handler Service Name	Specifies the name of the service which is configured with Salesforce Connector.
Retry Count	Specifies the maximum number of times that the system should attempt for the recoverable error if the initial attempt fails. The default is 3.
Callback Service Name	Specifies the name of the callback service in Salesforce Connector to perform logging. **Note:** You can also customize the callback service and provide it in this field.
Run As User	Click  to assign a user to a selected listener. By default, **Run As User** is set to Administrator. You can now configure the **Run as User** in the Integration Server Administrator to assign a user to a listener.

Example of an error response object that Salesforce sends to its subscribed clients

If a connection is lost due to unexpected network disruption, Salesforce server times out the client and deletes the client state. The client attempts to reconnect but the connection is rejected with the *403::Unknown Client Error* error response object because the client state does not exist anymore. The error response returned when the client attempts to reconnect after a timeout is as follows:

```
{
          "error":"403::Unknown client",
          "successful":false,
          "advice":{"interval":0,"reconnect":"handshake"}
         }
```

When the client receives the *403::Unknown Client Error* with the `"reconnect":"handshake"` advice field, the client must perform a new handshake. If the handshake is successful, the client must re-subscribe to the channel in the handshake listener.

Managing Cloud Connectors

After you publish a cloud connector, you must enable it as described below. In addition, you can:

- View its configured connection pool instance(s).
- View other details about the connector.
- Configure the properties of the connector as desired.

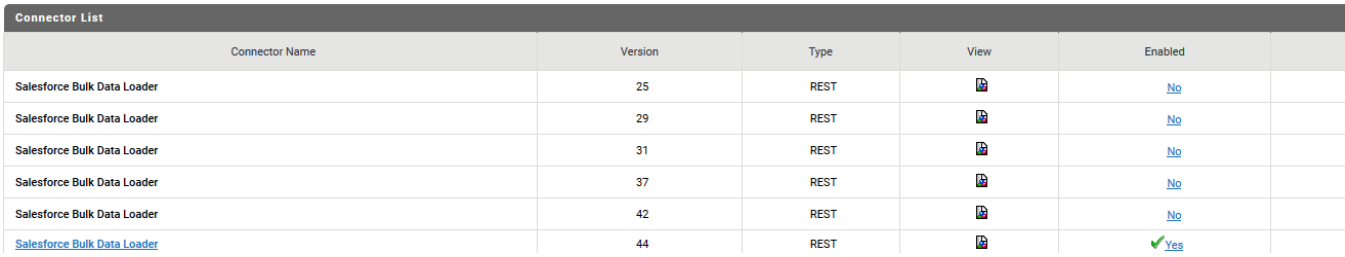
- Delete the connector.

Note:
When a new connector version is released, in the provider package, the new connector version is enabled, and all older connector versions are disabled. If you still want to work with an older connector version, you can manually enable the older connector version, if available, in the package. To manually enable a cloud connector, from Integration Server Administrator, go to **Solutions > CloudStreams > Providers**. Select a provider and on the **Connector List** screen, for the connector you want to enable, click on **No** in the **Enabled** column.

To manage a cloud connector

1. In the Integration Server Administrator, click **Solutions > CloudStreams > Providers > *your_provider_name***.

   The list of connector(s) that have been defined for that provider will appear in the **Connector List**.



2. You can do the following in the **Connector List** table:

Column	Description
Connector Name	Click the connector name to display the connector's configured connection pool instance(s). To create a new connection pool instance, click **Configure New Connection** and complete the fields on the Configure Connection page.
View	Click the icon to display details about the connector.
Enabled	Click **No** or **Yes** to enable or disable the connector respectively.
Delete	Click the icon to delete the connector. You must disable the connector before you can delete it.
Configure	Click the icon to display the **Connector Configuration** page, which displays the connector properties you can configure. You must disable the connector before you can edit the properties.

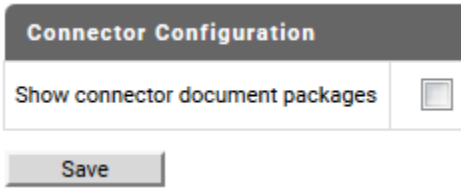
Managing Cloud Connector Packages

In some connectors, document types are shipped as a part of the provider package itself. When you install a provider package, CloudStreams can also **optionally** generate document types.

Note:
Any document type package generated is hidden by default. It is not recommended to modify a document type.

To view the document types in Software AG Designer

1. In Integration Server Administrator, click **Packages > Management**.
2. In the **Package List** table, you can enable/disable the packages, reload them, archive them, or delete them.
3. Select **Solutions > CloudStreams...**
4. On the CloudStreams screen, click a provider from the **Providers** section.
5. On the **Connectors** screen, click **Yes** in the **Enabled** column for that connector. For a disabled connector, click **Configure**.



6. In the **Configure Connector** screen, select the **Show connector document packages** option and click **Save**.
7. Go to the **Service Development** perspective in Software AG Designer.

   The document types will be visible in the **Service Development** perspective under the selected connector package.

Using inline connection details while running a cloud connector service

CloudStreams allows you to run a cloud connector service by passing connection details that are different from the connection details configured in the connection configuration page in webMethods Integration Server Administrator.

While designing a cloud connector service, select the **Use inline connection** option to enable the **Connection** panel. Then select the fields you want to include in the cloud connector service input signature in the **Connection** panel. When you select the fields, the cloud connector service input signature will be automatically updated based on your selections. Only the fields that are part of the signature gets mapped as the input request. You can override their values at run time. Default values will be used for other fields for creating the connection. The details specified in the **Connection** panel takes precedence over the details specified in the connection configuration page in webMethods Integration Server Administrator.

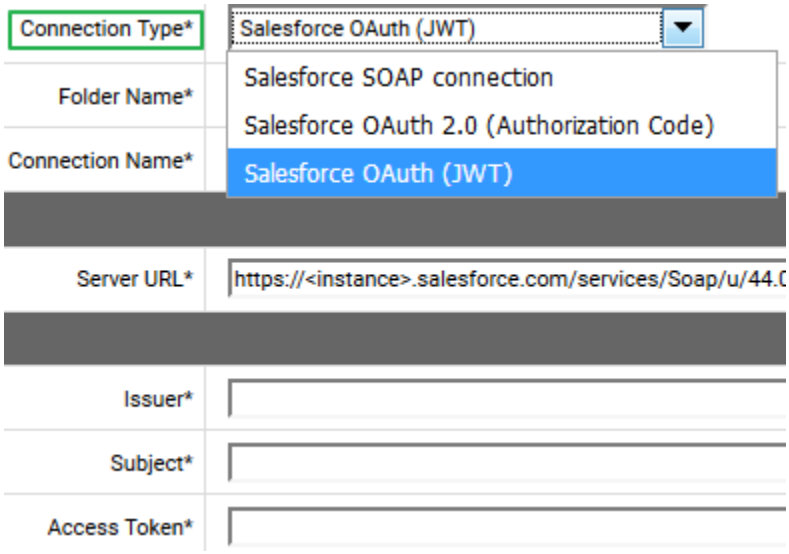
Note:
While running the cloud connector service, if you have provided both the connection alias and the inline connection details, CloudStreams reads the data from the connection alias and merges the connection alias data with the data provided in the inline connection, without modifying the existing configuration of the connection alias.

See the *Editing a Cloud Connector Service for a REST-Based Provider* or *Editing a Cloud Connector Service for a SOAP-Based Provider* sections in the *webMethods Service Development Help* for more information.

Support for multiple authentication schemes in a single connector

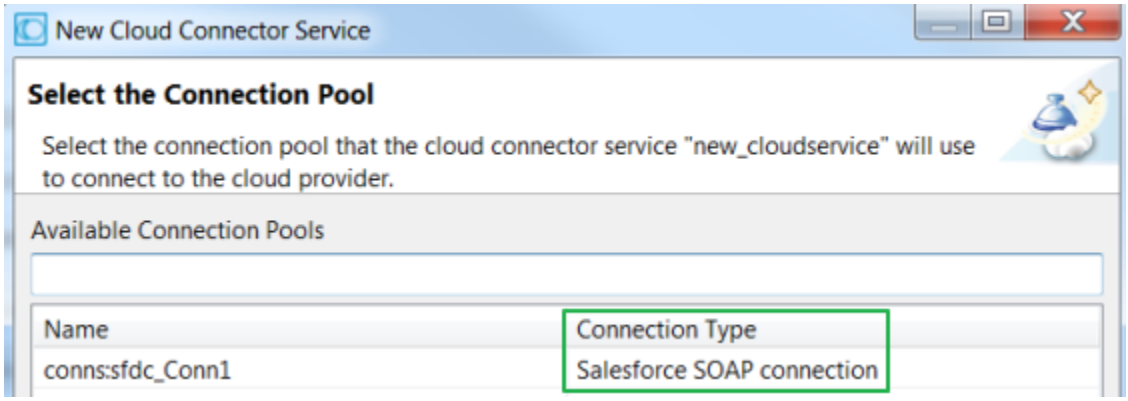
In CloudStreams v10.3 and earlier releases, while creating a connector, you could associate only one connection type or authentication scheme with a connector. As some SaaS providers such as Salesforce CRM v44 offer multiple authentication options to access their APIs, the connector developer needed to create different connectors for different authentication schemes, for example, one connector for basic authentication and another connector for OAuth 2.0 authentication.

From the CloudStreams v10.4 release, a connector developer can build a single connector that supports multiple authentication schemes. After the connector that supports multiple authentication schemes is built, the connector user can select different connection types (authentication schemes) for the same connector and for the same back end from the connection configuration page in webMethods Integration Server Administrator.



The **Connection Type** field lists all the connection types based on the supported authentication schemes by the connector and different connection configuration fields appear based on the selected **Connection Type**. The **Connection Type** appears on the **Connections** page while editing, viewing, or copying the connection.

While creating a cloud connector service or a connector listener of a connector that supports multiple authentication schemes, you will be able to view all the connections and their corresponding connection type in the **Select the connection pool** window in the Service Development perspective in Software AG Designer.



The connection type value in Software AG Designer, for example, **Salesforce SOAP connection**, is the same value as selected in the connection configuration page in webMethods Integration Server Administrator while creating a new connection.

Generating OAuth 2.0 Tokens while configuring connections

In CloudStreams v10.3 and earlier releases, you could not generate Access Tokens required to configure the connection for OAuth 2.0 authorization. From the CloudStreams v10.4 release, for some connectors such as Salesforce CRM v44, you can generate the OAuth 2.0 Access Token while

creating a new connection or while editing an existing connection from the connection configuration page in webMethods Integration Server Administrator.

Approaches for generating OAuth 2.0 tokens

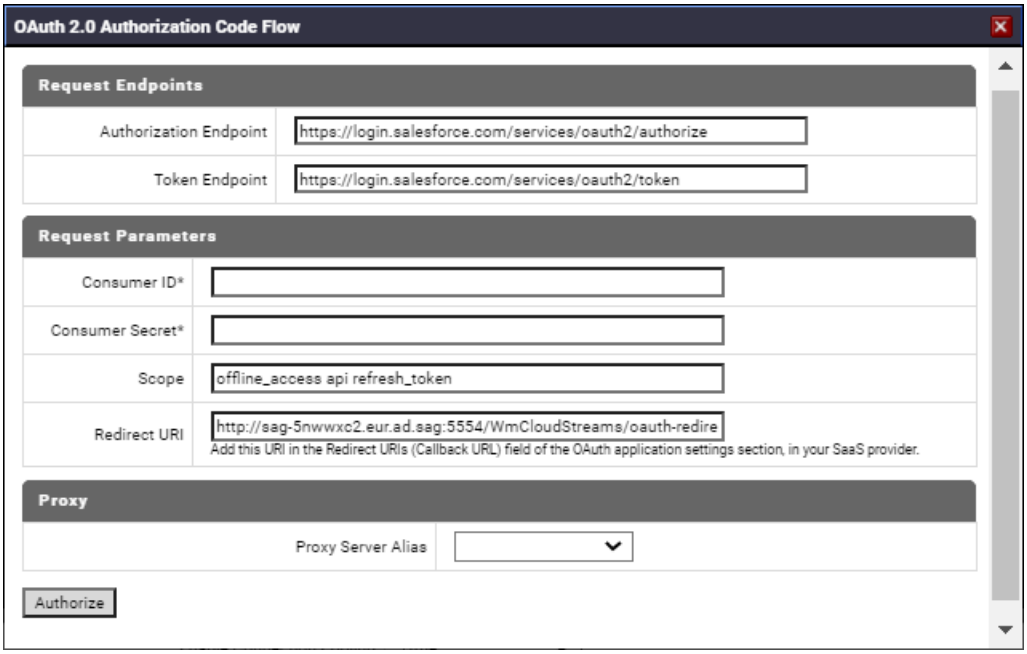
- **Authorization Code Flow** – An OAuth 2.0 flow that is used to grant an access token to server-hosted applications. In this flow, CloudStreams opens the web page for authorization, and after authorization, redirects the authorization code to CloudStreams.
- **JSON Web Token (JWT) Flow** – Authorization by a signed JWT.

Prerequisites

- Install CloudStreams Server v10.4 or a later version with the latest fix level.
- Install the supported v10.4 CloudStreams Provider, for example, v10.4 WmSalesforceProvider v44.
- Enable SaaS provider settings. For example, for Salesforce v44, you must have the Salesforce.com Connected App with Digital Certificate created by the keystore configured in webMethods Integration Server Administrator and enabled the OAuth settings. For information on how to configure Integration Server keystores, see the *Securing Communications with the Server* section in the *webMethods Integration Server Administrator's Guide*.

Generating OAuth 2.0 tokens

1. To generate tokens using the **Authorization Code Flow**, on the **Configure Connection** page in webMethods Integration Server Administrator, click **Generate Access Token**. The **OAuth 2.0 Authorization Code Flow** dialog box appears.



2. On the **Request Endpoints** section, provide the URL for **Authorization Endpoint** and **Token Endpoint**.

3. Obtain the required **Request Parameters** from the SaaS provider application, for example, OAuth App Consumer ID (Client ID), Consumer Secret (Client Secret), and Scope. The fields in the **Request Parameters** section may vary as per the SaaS provider.

4. In the **OAuth 2.0 Authorization Code Flow** dialog box, type the required request parameters you obtained in the previous step. If the request must be sent through a proxy server, in the **Proxy Server Alias** field, specify the alias name of the enabled proxy server configuration on webMethods Integration Server that will be used to route the request. Add the same **Redirect URI** in the Redirect URIs (Callback URL) field of the OAuth application settings section in your SaaS provider.

5. Click **Authorize**.

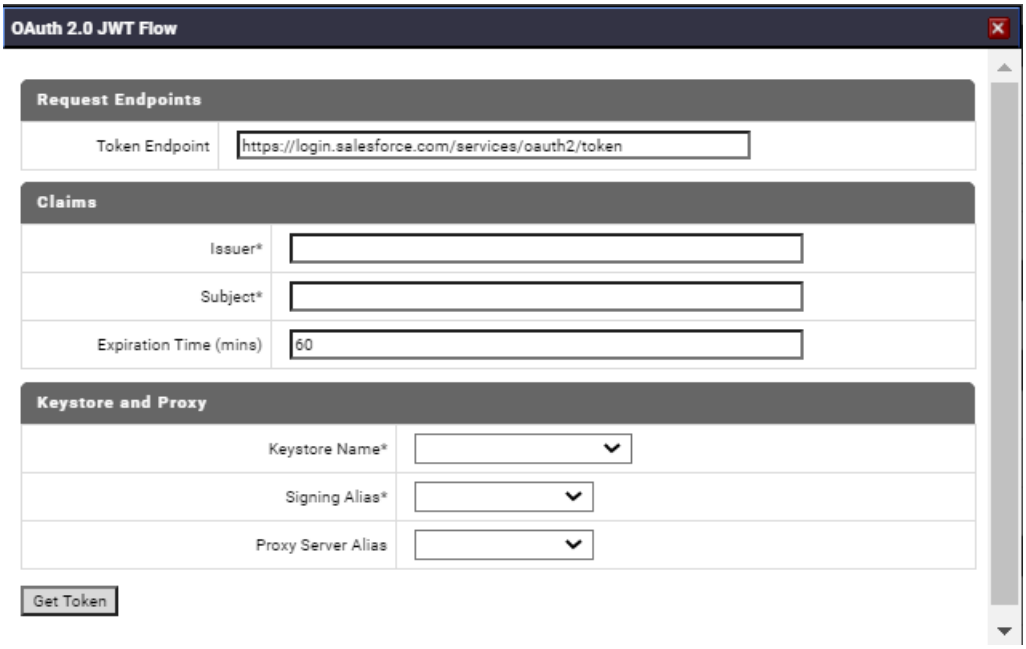
   The authorization URI is built and the authorization page of the SaaS provider opens.

6. Log in to the SaaS provider and grant access to the required scopes, if any. If you have a developer account, you already have a default authorization server created for you. The authorization server is the server that issues the access token.

   After successful authorization, the authorization server redirects the authorization code to CloudStreams.

7. CloudStreams exchanges the authorization code for an access token with the SaaS provider. The fields in the **Connection Groups: OAuth V2.0 (Authorization Code Flow)** section are populated after a successful response from the SaaS provider.

8. To generate tokens using the **JSON Web Token (JWT) Flow**, on the **Configure Connection** page in webMethods Integration Server Administrator, click **Generate Access Token**.



9. For the **Request Endpoints** section, provide the URL where the token resides in the **Token Enpoint** field.

10. For the **Claims** section, obtain the Claims information from the SaaS provider application, for example, **Issuer** and **Subject**. The fields in the **Claims** section may vary as per the SaaS provider.

11. Type the following required claims you obtained in the previous step:

    - **Issuer** - Client ID, or Identifier, or name of the server or system issuing the JWT token.
    - **Subject** - Identifier or the name of the user this token represents.

12. In the **Claims** section, type the following details:

    - **Expiration Time (mins)** - Time after which the JWT expires.

13. In the **Keystore and Proxy** section, type the following details:

    - **Keystore Name** - The Integration Server keystore that CloudStreams should use. This field lists all available Integration Server keystores. If there are no configured Integration Server keystores, the list will be empty. For information on how to configure Integration Server keystores, see the *Securing Communications with the Server* section in the *webMethods Integration Server Administrator's Guide*.
    - **Signing Alias** - This alias is the value that is used to sign the outgoing request from CloudStreams to the authentication server. It is auto-populated based on the keystore selected in the **Keystore Name** field. This field lists all the aliases available in the chosen keystore. You must provide a signing alias to sign the JWT payload.
    - **Proxy Server Alias** - If the request must be sent through a proxy server, in the **Proxy Server Alias** field, specify the alias name of the enabled proxy server configuration on webMethods Integration Server that will be used to route the request.

14. Click **Get Token**.

    The generated JWT token will be sent to the authentication server and the fields in the **Connection Groups: OAuth V2.0 (JWT Flow)** section will be populated after a successful response from the SaaS provider.

Multipart/form-data content type

Though application/x-www-form-urlencoded is a more natural way of encoding, it becomes inefficient for encoding binary data or text containing non-ASCII characters. The media type *multipart/form-data* is the preferred media type for request payloads that contain files, non-ASCII, and binary data. CloudStreams supports this media type using which you can embed binary data such as files into the request body in a transparent way.

A multipart/form-data request body contains a series of parts separated by a boundary delimiter, constructed using Carriage Return Line Feed (CRLF), "--", and also the value of the boundary parameters. The boundary delimiter must not appear inside any of the encapsulated parts.

Each part must contain a Content-Disposition header field where the disposition is *form-data*. The Content-Disposition header field must also contain an additional parameter of *name*. For form data that represents the content of a file, a name for the file should be supplied as well, by using a *filename* parameter of the Content-Disposition header field.

Each part may have an optional *Content-Type* header field which defaults to *text/plain*. If the contents of a file are to be sent, the file data should be labeled with an appropriate media type, if known, or *application/octet-stream*.

Example of a multipart request

```
--BOUNDARY
Content-Type: application/json
Content-Disposition: form-data; name="job"
{
  "object":"Contact",
  "contentType":"CSV",
  "operation":"insert"
}
--BOUNDARY
Content-Type: text/csv
Content-Disposition: form-data; name="content"; filename="content"
(Content of your CSV file)
--BOUNDARY—
```

Parts for a REST resource

CloudStreams defines the following three part types:

Part Type	Description
TEXT	Represents a simple text part of a multipart/form-data payload where the content of the part is of type text/plain. You can use this part to send raw text data as the content of a part, in a multipart request. From a file upload perspective, this part is generally used to convey extra information about the upload behavior, like target folder paths/folder names where the file has to be uploaded.
FILE	Represents a binary part of a multipart/form-data payload where the content of the part is binary or the content of the file itself. To upload a file, you will normally use this part. For file upload kind of use cases, this part is the main or mandatory part required by the service provider. **Note:** There is no limit in terms of the size of the file part that you can send as part of multipart. It depends on the available heap space.
DOCUMENT	Represents a part where the content of the part is of type application/json or application/xml.

Defining the parts for a REST resource

You can define the parts for a REST resource in the following ways:

- Connector XML
- Service Development perspective in Software AG Designer

Using the Connector XML to define the parts for a REST resource

If you are a connector developer and are sure about all combinations of all the parts for a REST resource, add the part definitions under *Resource Definitions* in the `Connector XML` file. This reduces your effort while developing cloud connector services as you can just create and run the cloud connector services and there is no need to add any parts by using the Service Development perspective in Software AG Designer. In the `Connector XML` file, if the *isRequired* attribute of the part element is set to *true*, the part will be available in the service signature. If *isRequired* is marked as *false*, the part appears in the *Attachment* table but does not appear in the signature. You can still add a new part by using the Service Development perspective in Software AG Designer.

- **TEXT** - In this definition, the following table describes the various attributes of the part element:

Attribute Name	Description
name	Name of the part.
contentType	Content type of the part's content.
Type	Part type, for example, here it is TEXT.
defaultValue	If provided, sets as the value of this text part.
isRequired	Represents whether this part will be included in the signature.

The following example shows how the file uploading path, `sampleFolder`, is sent using a text part.

Note:
Set the `messageFormatterType` attribute of the request element as `multipart/form-data`. This will denote this resource as a multipart form-data resource.

```
<request name="UploadFileRequest" messageFormatterType="multipart/form-data"
builderType="application/octet+idataoref">
   <parts>
        <part name ="folder_paths" contentType ="text/plain" type="TEXT"
           defaultValue="sampleFolder" isRequired="true"/>
        </parts>
</request>
```

- **FILE** - In this definition, the following table describes the various attributes of the part element:

Attribute Name	Description
name	Name of the file part.
contentType	Content type of the file that is being uploaded.
Type	Part type, for example, here it is FILE.
defaultValue	If provided, sets as the filename by which it will be uploaded.
isRequired	Represents whether this part will be included in the signature.

The following example shows how a raw file *takeit.txt* having content type *text/plain* is sent using a file part.

```
<request name="UploadFileRequest" messageFormatterType="multipart/form-data"
builderType="application/octet+idataoref">
  <parts>
     <part name ="files" contentType ="text/plain" type="FILE"
     defaultValue="takeit.txt" isRequired="true"/>
  </parts>
</request>
```

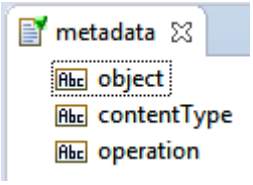
- **DOCUMENT** - In this definition, the following table describes the various attributes of the part element:

Attribute Name	Description
name	Name of the file part.
contentType	The serialization type to which the document will be serialized and become the part content. CloudStreams supports only `application/json` and `application/xml` as the content type for this part.
Type	Part type, for example, here it is DOCUMENT.
defaultValue	If provided, sets as the document reference.
isRequired	Represents whether this part will be included in the signature.

Example: You want to send the following JSON payload, which is of type `application/json` as the part's content.

```
{
  "object":"Contact",
  "contentType":"CSV",
  "operation":"insert"
        }
```

To transmit this payload in a part, you have to create an Integration Server document, for example, *metadata*, which will contain three string fields.



In this example, the fields are object, contentType, and operation. You have to point the document namespace as the default value, in this example, *multipart.sample:metadata*, and the content type as *application/json*. CloudStreams can now serialize the Document and make it available as the content of the document part.

The following is a sample part definition for the Document part:

```
<request name="UploadFileRequest" messageFormatterType="multipart/form-data"
builderType="application/octet+idataoref">
   <parts>
      <part name ="job" contentType ="application/json" type="DOCUMENT"
      defaultValue="multipart.sample:metadata" isRequired="true"/>
        </parts>
</request>
```

Using the Software AG Designer to define the parts for a REST resource

This approach allows the service developer to add parts, after creating the cloud connector service. The **Attachment** panel in the **Service Development** perspective in Software AG Designer lists all the parts that have been added.

- **TEXT** - In the **Name** field, type the name of the file part as documented in the SaaS provider API documentation. The **Name** field is mandatory. In the **Value** field, type the value of the part. If you do not give any value, an empty string will be set as the default value.
- **FILE** - In the **Name** field, type the name of the file part as documented in the SaaS provider API documentation. The **Name** field is mandatory. In the **Content Type** field, type the content type of the file you are uploading. If you do not give any value, the default value will be set as application/octet-stream. In the **File Name** field, type the file name by which you want the file to be uploaded. If you do not give any value, the default value will be set as *attachment*.
- **DOCUMENT** - In the **Name** field, type the name of the file part as documented in the SaaS provider API documentation. In the **Content Type** field, type the content type of the body part that you want your document to be serialized and sent to the back end. CloudStreams supports only `application/json` and `application/xml` as the content type for this part. For the **Document** field, first create a document that matches the payload and then select that document. The **Name**, **Content Type**, and **Document** fields are mandatory.

Note:
All options including the **Add Part** option are disabled if the resource is not of type multipart/form-data. Currently, multipart/form-data payload is supported only at the request level, that is, only in the input signature.

If you do not provide the optional values, CloudStreams sends the default values as mentioned in the following table to the back end.

Part Type	Field Name	Default Value
TEXT	Value	Empty string
FILE	Content Type	application/octet-stream
FILE	File Name	attachment

See the *Building Cloud Connector Services* section in the *webMethods Service Development Help* for more information on how to add parts using Software AG Designer.

Message Transmission Optimization Mechanism (MTOM)

MTOM is a specification that enables clients to exchange binary data with Web service providers in an optimized way. MTOM is used in conjunction with XOP (XML-binary Optimized Packaging) specification.

CloudStreams allows you to send binary data as a Base64 encoded format, using the MTOM specification.

If you are a connector developer, and want to make a SOAP operation (that is involved in sending the binary data) MTOM enabled, set the *enableMtom* property to *true* in the `Connector XML` file:

```
<operation …>
 <properties>
  <property name="enableMtom" value="true" type="boolean"/>
 </properties>
</operation…>
```

When the *enableMtom* property is set to *true*, the message contains additional MIME parts as per the MTOM specification, which are referred to in the SOAP message body.

If you are a connector user, in Software AG Designer, do the following to exchange binary data using the MTOM specification:

- Use the *pub.file:getFile* service to retrieve the file as binary data.
- Use the *pub.string:base64Encode* service to encode the binary data as a Base64 string.
- Map the encoded string to a field in a cloud connector service, the Content Type of which is base64Binary.

Note:
See the *webMethods Integration Server Built-In Services Reference* for information on the *pub.file:getFile* and *pub.string:base64Encode* services.

Service signatures having complex nested document structures

For some back ends, the service signatures are very complex in nature. If those signatures are represented as document types, the document structures become very nested and deep. If such document structures are used to create service signatures, out of memory issues are observed. Therefore for such back ends, connector services could not be created.

In earlier releases, CloudStreams always concretized the document types that are referred to in the service signature and made a copy of the referred documents in line into the service signature. With this approach, service signatures could not be created for back ends with complex and deep signature structures, as loading those huge and nested document types resulted in out of memory issues.

For those types of back ends, you must use the *Hybrid* approach for creating connector service signatures. In this approach, a service signature is generated by combining reference of document types with concrete document types (optional). This approach only concretizes the signature path, which requires a hint to be applied. A hint is denoted by declaring a literal to XPath mapping, where the literal value is the hint that is intended to be applied on the end element of the XPath.

Sample Literal-XPath mapping

For example, if the following literal to XPath mapping is defined in the operation/resource definition, CloudStreams will concretize all the reference documents that occur in the given XPath mapping path, which is *tns:AllowFieldTruncationHeader/tns:allowFieldTruncation*.

```
<mappings>
<map context="IN" isRequired="true" isFixed="false" displayName="Field
Truncation Header" name="allowFieldTruncation">
<description>If this header value is true then it truncates data if the data
is too long and does not fit into the given field</description>
<source type="LITERAL">false</source>
<target type="XPATH" documentRef="wmSalesforceConnector_v31.doctypes:
Header_AllowFieldTruncationHeader">
<xpath>tns:AllowFieldTruncationHeader/tns:allowFieldTruncation</xpath>
</target>
</map>
</mappings>
```

Rest of the signature that does not have any hint to be applied will be rendered as a reference only. This helps you to reduce the foot print of the signature. In the absence of any hint or literal to XPath mapping, this approach leads to a service signature full of only document references.

To adopt the hybrid approach for connector service generation, you must make the following entry in the operation/resource level to create the signature:

```
<properties>
      <property name="allowNSReferenceInSignature" value="true"/>
</properties>
```

Further, you must also configure the **Document Expansion** preferences with the recommended values mentioned in the *Document Expansion Preferences* section in the *webMethods Service Development Help*.

Note:
This approach is not applicable for back ends where business object look ups are involved, for example, Salesforce CRM.

Multiple root JSON data format with anonymous collection

The multiple root feature enables CloudStreams to support JavaScript Object Notation (JSON) APIs which return as well as accept anonymous collection, that is, collection without a key, of JSON objects provided or accepted by a JSON provider. CloudStreams supports anonymous JSON object collection in the request and response design time.

You can use CloudStreams to access those APIs which return as well as accept anonymous collection of JSON objects for a JSON backend. The number of APIs supported now by CloudStreams is more. Most of the social connectors support anonymous collection in most of the APIs.

This feature also helps you to get error messages for scenarios where the exclude root is marked as `true`. You can experience a broader connectivity to all sets of JSON backends, especially social connectors.

The multiple root feature allows you to consume all types of APIs and does not restrict you to a subset of APIs. You can create cloud services that communicate with the JSON backends, which expect or accept an anonymous collection from the backend. With added support in the Software AG Designer, you can now create document types of the JSON objects whose collection is expected or accepted from the JSON backend and can also map it as a logical collection in the design time of the cloud service. CloudStreams automatically handles root less array of JSON objects at runtime and the contents of each object separately.

Note:
Response having multiple root JSON data format with anonymous collection is supported from version 9.8 and later versions of Software AG Designer and CloudStreams Server.

Request having multiple root JSON data format with anonymous collection is supported from version 9.9 Fix 2 and later versions of Software AG Designer and CloudStreams Server.

REST Parameters

CloudStreams REST connector services allow you to set parameters which become part of the outgoing request.

REST services rely on HTTP methods (GET, POST, PUT, DELETE) to make requests to a SaaS provider. Thus the parameters are closely tied to these HTTP methods, as they are sent as part of these HTTP method requests.

The parameters are typically part of the HTTP URI which may be of the following format:

```
[scheme:][//authority][path][?query]
```

Parameter Types

CloudStreams supports the following parameter types or styles:

Type	URI Scope
"URI_CONTEXT" on page 223	path
"QUERYSTRING_PARAM" on page 223	query
"CFG_PARAM" on page 224	authority, path
"FORM_ENCODED_PARAM" on page 224	body

URI_CONTEXT

URI_CONTEXT parameters are passed as the path component of a REST resource URI, and the parameter names correspond to the URI path variable names specified in the {} annotation.

Example

A sample request is as follows:

```
services/async/25.0/job/{jobId}
```

In the above sample request, the URI path variable name `jobId` is specified as a parameter to the job resource. The annotation {} is set to the variable name `jobId`. At service run-time, the variable is substituted with it's runtime value to form the dynamic path.

A sample resource definition with URI_CONTEXT parameter is as follows:

```
<resource name="Job" method="POST" path="services/async/25.0/job/{jobId}">
    <parameters>
       <parameter name="jobId" isRequired="true" style="URI_CONTEXT"/>
    </parameters>
</resource>
```

QUERYSTRING_PARAM

QUERYSTRING_PARAM parameters are passed as the query component of a REST resource invocation request.

Example

The following example demonstrates a typical HTTP GET request with parameters that form a query string of the resource URI.

```
GET  /status?key1=value1&key2=value2 HTTP/1.1
Host: www.softwareag.com
Content-Length: 0
```

Note that the parameters are added to the path after a "?", and specified as ampersand (&) separated list of key-value pairs, with the corresponding Content-Length set accordingly. The key & values passed as parameters are URL encoded by CloudStreams.

A sample resource definition with QUERYSTRING_PARAM parameter is as follows:

```
<resource name="GetStatus" method="GET" path="/status">
```

```
    <parameters>
       <parameter name="key1" isRequired="true" style="QUERYSTRING_PARAM"/>
       <parameter name="key2" isRequired="false" style="QUERYSTRING_PARAM"/>
    </parameters>
</resource>
```

CFG_PARAM

CFG_PARAM parameters signify an internal contract between the connector tier and the connecter-specific authentication scheme along with the virtual runtime layer.

The allowed set of parameter name, which can be specified as CFG_PARAM depends on the authentication scheme used for the connector definition.

Example

`aws.bucketName` is a parameter, which is used by the Amazon Authentication Scheme version 3, for specifying the dynamic host based on the parameter value.

While executing a resource with such parameter, the service endpoint is prefixed with the appropriate bucket name.

```
<resource name="GetBucket" method="GET" path="/">
    <parameters>
        <parameter name="aws.bucketName" isRequired="true" style="CFG_PARAM"/>
     </parameters>
</resource>
```

FORM_ENCODED_PARAM

FORM_ENCODED_PARAM allows you to send simple key value parameters embedded in the Request body for POST or PUT requests. This uses the default web form encoding, which is *application/x-www-form-urlencoded*.

You can define a parameter of type FORM_ENCODED_PARAM if you want to send the parameter as part of the Request body. You can also format the input data into a desired output format using the formatter, similar to QUERYSTRING_PARAM.

To define a form URL encoded parameter, you have to create a parameter type and set the style as FORM_ENCODED_PARAM

Example

A sample resource definition with FORM_ENCODED_PARAM parameter is as follows:

```
<resource method="POST" path="/Accounts" name="CreateSubAccount"
           displayName="Create SubAccount">
            <description>Create new subaccount</description>

            <parameters>
               <parameter name="FriendlyName" isRequired="true"
               style="FORM_ENCODED_PARAM">
                 <description>Friendly Name</description>
               </parameter>
               <parameter name="Status" isRequired="true"
```

```
                style="FORM_ENCODED_PARAM">
                  <description>Status of account</description>
                </parameter>


            </parameters>

            <responses>
                ...
            </responses>
        </resource>
```

For passing parameters of FORM_ENCODED_PARAM type, you must not define the Request body for a Resource, as the generated parameter key value string will be automatically embedded in the Request body. In the above example, two parameters of type FORM_ENCODED_PARAM are defined as *FriendlyName* and *Status*. When the service created by the sample resource definition is run, the following sample Request payload is sent to the back end:

```
POST /2010-04-01/Accounts/AC53e967de85059b65475d0059882a9e02 HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 39
Host: api.twilio.com
Connection: Keep-Alive
Authorization: Basic ********************************
```

FriendlyName=NewFrName&Status=suspended is also sent as part of the same Request body:

```
FriendlyName=NewFrName&Status=suspended
```

In the above sample Request payload, as the Content-Type is automatically set to *application/x-www-form-urlencoded*, you do not have to explicitly set the Content-Type. Further, the parameters, *FriendlyName* and *Status* are URL encoded and are specified as an ampersand (&) separated list of key-value pairs (*FriendlyName=NewFrName* and *Status=suspended*), with the corresponding Content-Length set accordingly.

Parameter Data Types

CloudStreams currently supports the data types String and Record.

String

String data type represents a simple parameter which is an individual string key-value parameter. This is the most commonly used data type and the default type, if not specified explicitly.

Record

Record data type handles complex parameters involving a collection of related key-value pairs. These parameters can have indices's for one or more parameter entry. Such parameters can ideally be represented with an IS document type, and configured as a “Record” data type.

Example

Consider a complex parameter structure as follows:

```
+ Filter[]
    - Name
    + Value[]
```

This needs to be represented as a query string like:

```
Filter.1.Name=instance-type&Filter.1.Value.1=m1.small&Filter.1.Value.2=t1.micro
```

Such a parameter can be defined by creating an IS document type, and referring it as a “Record” data type parameter within the resource definition.

CloudStreams provides the capability to represent and transform such complex parameters into the desired format using its predefined formatter implementation. The resource's parameter definition can be enhanced to refer to a formatter, as follows:

```
<parameter name=”Filter” dataType=”Record”
documentRef=”documents.paramType:FilterList”
style=”QUERYSTRING_PARAM”>
    <formatter service=”wm.cloudstreams.service.util.formatters:paramFormatter”
type=”paramFormatter”/>
</parameter>
```

At runtime, the resource PATH would look like:

```
GET /?Filter.1.Name=instance-type&Filter.1.Value.1=m1.small&Filter.1.Value.2=t1.micro
```

For more details on formatters and customizing the formatter behavior, see “Parameter Formatters for REST Connector Services” on page 226.

Parameter Formatters for REST Connector Services

CloudStreams provides a parameter formatter which enables you to format input data into a custom format as desired.

Alternatively, in cases where the default parameter formatter is not sufficient to achieve the desired parameter behavior, you can write a custom implementation for the parameter formatter.

The input will typically be data present in a complex structure, for example, an IS document type.

The output will be data organized in a desired output format, for example, a URL query string. The format in which the output data is generated depends on the implementation of the formatter.

Typically, you will use the QUERYSTRING_PARAM or the FORM_ENCODED_PARAM parameter types to format the input data into a desired output format, for example, a URL query string in case of QUERYSTRING_PARAM or in case of FORM_ENCODED_PARAM, a formatted query string in the Request body.

In order to form the query string, you can write a custom formatter service by implementing the bundled IS specification wm.cloudstreams.service.common.lookup.specs:formatterSpec.

At run time CloudStreams will give control to the formatter service to format the data appropriately and to return the formatted string data. Then, CloudStreams will encode the data during a post-processing step, and the final query string will be formed.

For details, see:

Using the Default Parameter Formatter

For convenience, CloudStreams provides a default formatter implementation that handles the most common formatter usage:

wm.cloudstreams.service.util.formatters:paramFormatter

This formatter allows you to specify a complex IS document type, and to implement the desired output. The input will typically be data in a complex structure, for example, an IS document type. The output will be a data organized in a desired output format, for example, a URL query string. The format in which the output data is generated depends on the implementation of the formatter. Typically, the formatter is used in conjunction with the QUERYSTRING_PARAM parameter type.

Example of using the default parameter formatter

Consider a SaaS back-end expecting the following parameter:

```
BlockDeviceMapping.n.DeviceName
BlockDeviceMapping.n.VirtualName
BlockDeviceMapping.n.Ebs.NoDevice
BlockDeviceMapping.n.Ebs.VolumeSize
```

where `n` represents a number.

1. Create IS document types for the above parameter definition.

   The IS document structure would look like this:

   ```
   BlockDeviceMapping[]
     + DeviceName
     + VirtualName
     + Ebs
         + NoDevice
         + VolumeSize
   ```

2. Set the `dataType` of the parameter to `Record`.

3. Set the document reference (`documentRef`) to the newly created IS document type in step 1.

4. Specify a formatter element within the parameter element, along with the service name (*service* attribute) and type (*type* attribute).

5. In case the implementation contains supported arguments, specify any customization by adding the appropriate argument name and value.

   The parameter definition for a resource using the above, within the Connector Descriptor, would look like this:

```
<parameter name="BlockDeviceMapping" dataType="Record"
documentRef="document.paramType:BlockDeviceList" style="QUERYSTRING_PARAM">
  <formatter service = "wm.cloudstreams.service.util.formatters:paramFormatter"
type = "paramFormatter">
    <arg name="startIndex" value="1" />
    <arg name="format" value="." />
  </formatter>
</parameter>
```

Note:
The default value for the type attribute within the scope of a parameter definition is “paramFormatter”. This type refers to the bundled IS Specification:

```
wm.cloudstreams.service.common.lookup.specs:formatterSpec
```

The generated representation of the above parameter would look like this:

```
BlockDeviceMapping.1.DeviceName=/dev/sdj&;BlockDeviceMapping
.1.Ebs.NoDevice=true&;BlockDeviceMapping
.2.DeviceName=/dev/sdh&;BlockDeviceMapping
.2.Ebs.VolumeSize=300&;BlockDeviceMapping
.3.DeviceName=/dev/sdc&;BlockDeviceMapping
.3.VirtualName=ephemeral1
```

The default formatter can be configured with the following supported arguments:

Argument	Default Value	Description
`startIndex`	0	Determines the index values of the keys e.g. should the n present in BlockDeviceMapping.n.DeviceName start with 0 or 1 or some number of your choice.
`format`	. (period)	Specifies how the document entries be separated. For example, BlockDeviceMapping.3.DeviceName is separated by a period (.).

Implementing a Custom Parameter Formatter

In cases where the default parameter formatter is not sufficient to achieve the desired parameter behavior, you can write a custom implementation for the parameter formatter.

Implement a custom formatter as follows:

1. Write an IS service implementation, adhering to the bundled IS specification wm.cloudstreams.service.common.lookup.specs:formatterSpec.
2. The fully-qualified service namespace should be referred with the formatter service attribute under the parameter definition. Ensure that the formatter type attribute is set to "paramFormatter".
3. Optionally, any service arguments can be specified within the formatter element definition.

Example

Consider a custom service implementation:

```
my.custom.formatter:sample
```

The parameter definition for a resource using the above, within the Connector Descriptor, would look like this:

```
<parameter name="SampleParam" dataType="Record"
documentRef="my.custom.document:sample" style="QUERYSTRING_PARAM">
  <formatter service = "my.custom.formatter:sample" type = "paramFormatter">
    <arg name="separator" value=";" />
  </formatter>
</parameter>
```

Viewing the Constraints Applied to Variables

Designer displays small symbols next to a variable icon to indicate the constraints applied to the variable as follows.

Variable	Constraint Status	Variable Properties
String	Required field.	The `Required` property is set to True.
String	Optional field.	The `Required` property is set to False.
String	Required field with content type constraint.	The `Content type` property specifies an IS schema or XML schema.
String	Optional field with content type constraint.	The `Required` property is set to False, and the Content type property specifies an IS schema or XML schema.
String	Required field with default value.	The `Fixed` property is set to False, and the defaultValue property specifies a default value. The variable has a default value, but you can override this default value with any other valid values while executing the service or mapping the variables.
String	Required field with fixed value.	The `Fixed` property is set to True, and the `defaultValue` property specifies a null value. The variable has a null value assigned to it by default and you cannot override this value. You cannot map this variable to another variable or assign any input values to this variable during service execution.
String	Required field with fixed default value.	The `Fixed` property is set to True, and the `defaultValue` property specifies a default value. The variable has a default value and you cannot override this value. You cannot map this variable to another

Variable	Constraint Status	Variable Properties
		variable or assign any input values to this variable during service execution.

Server Name Indication (SNI) Support

CloudStreams Server supports Server Name Indication (SNI) for Applications/Platforms websites. SNI allows multiple websites to exist on the same IP address. Without SNI, each hostname would require its own IP address for an SSL certificate to be installed. SNI solves this problem. CloudStreams provides SNI supported connections with configurable parameters such as **Enable SNI** and **SNI Server Name**.

- **Enable SNI** - Enable this option if the SaaS provider offers SNI-based TLS connectivity, and if you want to connect to an SNI enabled SaaS provider to send the host name specified in the **Server URL** field, as part of the TLS SNI Extension `server_name` parameter.
- **SNI Server Name** - If you want to explicitly specify a host name to be included as a part of the SNI extension `server_name` parameter, in case the host name is other than the host name specified in the **Server URL** field, specify the host name value in the **SNI Server Name** field.

For more information, see the documentation specific to your CloudStreams provider, for example, *webMethods CloudStreams Provider for Salesforce.com Installation and User's Guide*.

7 Clustering

Clustering extends the reliability, availability, and scalability of Integration Server and accomplishes this by providing the infrastructure and tools to use multiple Integration Servers as if they were a single virtual server.

CloudStreams supports the use of clustering by either distributing the processing load for streaming API events across Integration Server instances or limiting the processing to a single Integration Server node, while guaranteeing processing in each mode *exactly once*.

Apart from the infrastructure requirements enforced by Integration Server to operate in a clustered mode, CloudStreams leverages the distributed cache, which stores data in a Terracotta Server Array to coordinate processing of events between different Integration Server instances. See the *webMethods Integration Server Clustering Guide* for more information on clustering.

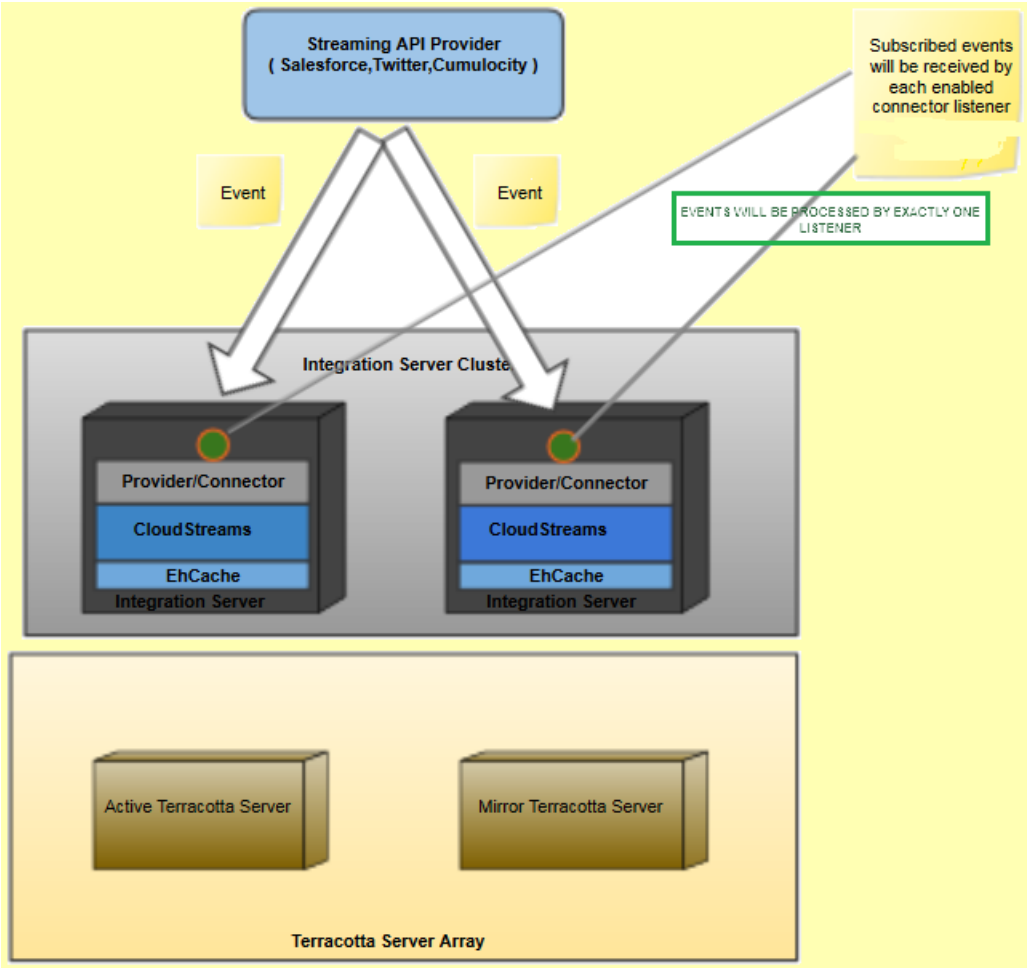
Clustering support for connector listeners

CloudStreams supports the use of clustering by either distributing the processing load for streaming API events across Integration Server instances or limiting the processing to a single Integration Server node, while guaranteeing processing in each mode *exactly once*.

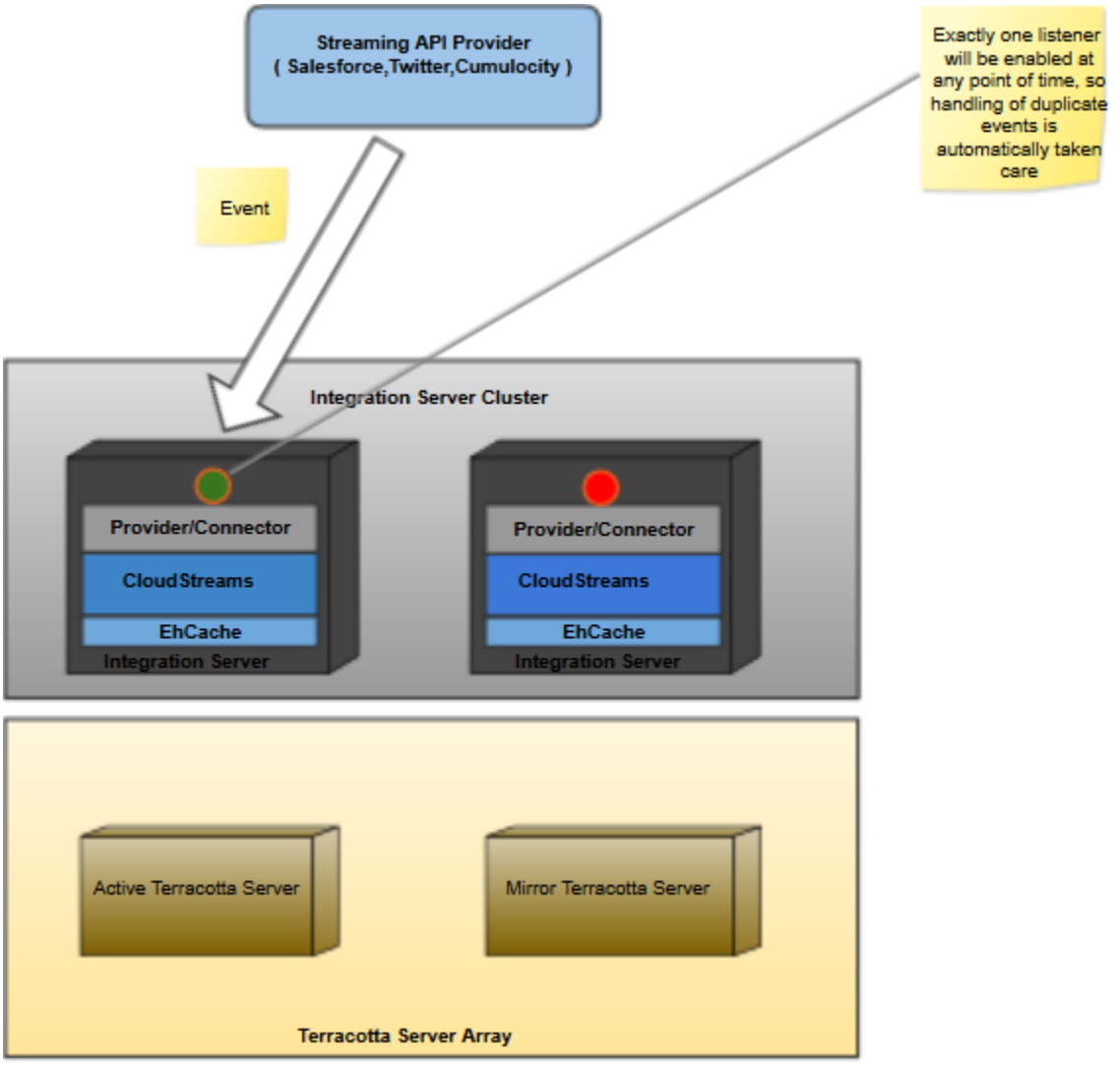
In v10.3 and earlier releases, CloudStreams connector listeners were not cluster aware. So when connector listeners were active on multiple nodes of an Integration Server cluster, each of them received and processed events independently as in a standalone Integration Server. This resulted in unwanted duplicate invocations of actions associated with the connector listeners. For example, if a connector listener is configured with the service invocation action and the same connector listener is replicated in an enabled state across other nodes of the Integration Server cluster, then the configured service invocation action is invoked on all the nodes of the cluster for each inbound subscribed event. From the CloudStreams v10.3 Fix 4 release, all connector listeners are cluster aware.

The following two cluster support modes are available for connector listeners:

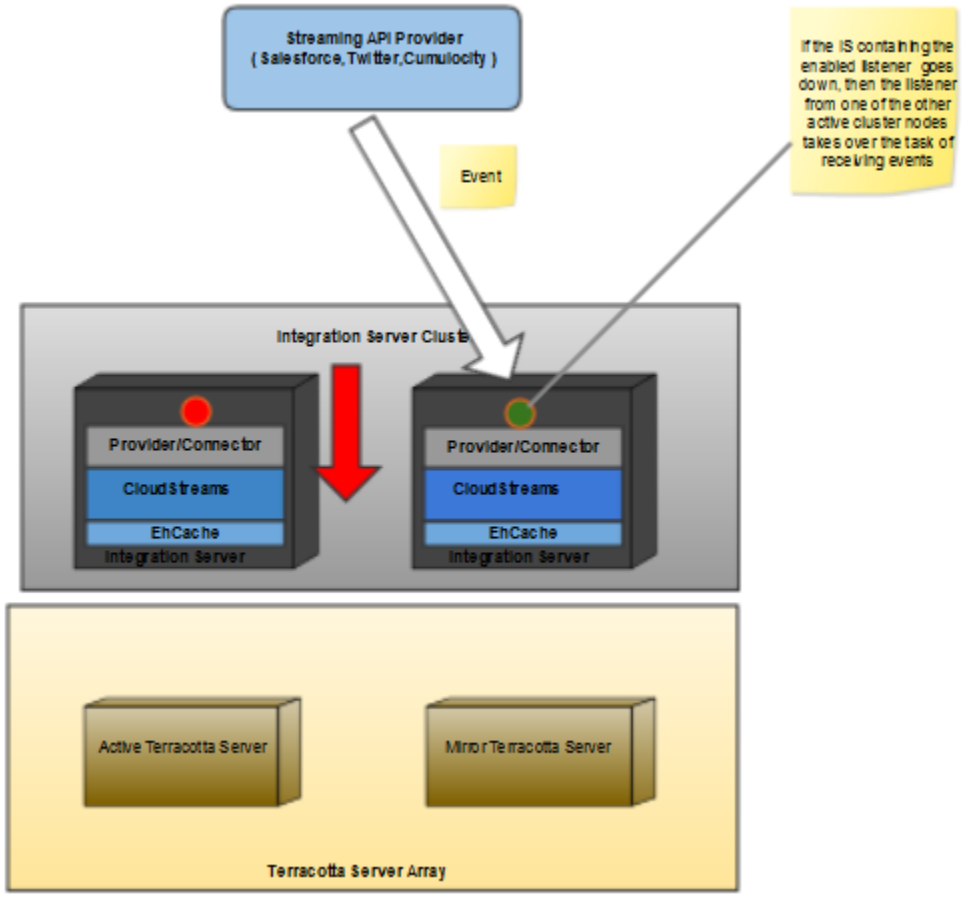
- **Multi-Node mode** - Any connector listener instance can process events, but for a given event, *exactly one* connector listener instance will process the events.



- **Single-Node mode** - A *designated connector listener* instance processes the events.



If the *designated connector listener* instance stops listening due to an Integration Server restart or any other reason, a different connector listener instance takes over the processing of the events and this connector listener instance will now be the designated connector listener instance.



For both modes, all instances of a connector listener will actively listen on all the deployed Integration Server nodes of a cluster.

Note:
The choice of a cluster support mode for a connector listener is predefined by the connector developer for a back end. The administrator or solution developer will not be able to select or change the cluster mode for a connector listener.

8 Using CloudStreams Configuration Variables Templates with webMethods Microservices Runtime

Overview

Microservices are independently deployable units of logic in which each microservice performs a single business function. Applications built in the microservices architectural style are developed as a suite of microservices.

Software AG offers a lightweight container called webMethods Microservices Runtime to host microservices you develop in Software AG Designer. Using Microservices Runtime, you can deliver microservices as an webMethods Integration Server package that includes a set of related services, interfaces, document types, and triggers that subscribe to topics or queues, or as a set of related packages of one kind, for example, five packages relating to Human Resources functions.

Microservices Runtime is optimized for execution in a Docker container and you can run a microservice or a set of related microservices in a Docker container. Docker images include configuration, which enables you to deploy the same configuration anywhere. The Docker image can include one package or a set of related packages.

Microservices Runtime provides the ability to create a Docker image from an installed and configured instance of Microservices Runtime. The Docker image contains the Microservices Runtime application including packages and Microservices Runtime assets. While a Microservices Runtime stores configuration information inside its file system and therefore inside any Docker image created from the Microservices Runtime, you can externalize and pass some configuration information to the Microservices Runtime at startup. You can accomplish this with Microservices Runtime by using a *configuration variables template* and environment variables.

Configuration Variables Template

A configuration variables template contains configuration properties that map to properties on the Microservices Runtime. You can set the property values externally in the template and then pass the values to a Microservices Runtime when it starts up. As part of the startup process, Microservices Runtime loads the information from the configuration variables template and replaces the configuration information stored in the file system.

By externalizing configuration information, you can use a single Docker image created for a Microservices Runtime across multiple environments, including different stages of the production cycle. For example, you might use different templates for specific environments such as testing versus production, or you might use the same template for all environments but use environment variables to vary the configuration in each environment.

A configuration variables template is a *properties file* that contains configuration data as a series of key-value pairs where the key name reflects the asset and particular asset property for which you can supply a value. For example,
`cloudstreamsconnection.AssetsPackage.xyzConnections\:s_conn1.connectionconfiguration.cn..connectTimeout=180000`
indicates that the *cloudstreamsconnection* alias named as *xyzConnections* under the namespace *xyzConnections\:s_conn1* is available in the package named *AssetsPackage*, and has a connection timeout of *180000*.

The configuration variables template contains key-value pairs for only a subset of the microservices container configuration, that is, the configuration variables template does not include key-value pairs for all of the configuration information for a microservices container. Only some configuration

information is supported for use with the configuration variables template. For example, for a OAuth token alias, the configuration variables template lists a key-value pair for the accessToken, accessTokenSecret, clientSecret, instanceUrl, refreshToken, and refreshUrl properties but the template omits key-value pairs for other properties, for example, the OAuth version.

When a configuration variables template is generated from an existing Microservices Runtime, the template is populated with configuration data from the Microservices Runtime instance. The template content reflects the Microservices Runtime configuration at the time Microservices Runtime generated the template.

Note:
For more information about microservices and configuration variables templates, see the *Developing Microservices with webMethods Microservices Runtime* document.

The following is an extract of the CloudStreams configuration variables template:

```
#Sample Generated Template
#Fri Aug 24 17:42:12 IST 2018
cloudstreamsconnection.AssetsPackage.xyzConnections\:s_conn1
.connectionconfiguration.cn..clientKeyAlias=
cloudstreamsconnection.AssetsPackage.xyzConnections\:s_conn1
.connectionconfiguration.cn..connectTimeout=180000
cloudstreamsconnection.AssetsPackage.xyzConnections\:s_conn1
.connectionconfiguration.cn..enableCompression=false
cloudstreamsdatabaseconfig.DatabaseConfig.pg.PgMenConfiguration
.perfDataEnabled=true
cloudstreamsdatabaseconfig.DatabaseConfig.pg.PgMenConfiguration
.reportInterval=100
cloudstreamsdatabaseconfig.DatabaseConfig.pg.jdbc.active=false
cloudstreamsdatabaseconfig.DatabaseConfig.pg.publish.events=false
cloudstreamslargefileconfig.LargeFileConfig.wst.largedata
.binaryStreamEnabled=true
cloudstreamslargefileconfig.LargeFileConfig.wst.largedata
.threshold=1234567
cloudstreamsoauthtokens.testOAUTH1.accessToken={AES}
LZx9vaF8gghTDI8YtM7Nxg\=\=
cloudstreamsoauthtokens.testOAUTH1.accessTokenSecret={AES}
tPtnGY6q6jsq/Kc3nqCQ5g\=\=
cloudstreamsoauthtokens.testOAUTH1.clientId=asdfdsfdsfds
cloudstreamsoauthtokens.testOAUTH1.clientSecret={AES}
eEE9SiDm81GM4FS6+1McAQ\=\=
```

Configuration Variables Template Assets

The following table identifies the CloudStreams assets and property names that correspond to the key names in the configuration template properties file.

Note:
Microservices Runtime generates an `application.properties` file that contains the key-value pairs from the applied template. The docker image must contain the same assets as available in the template file, so that the new values provided in the `application.properties` file can be applied while running the docker image.

Asset	Property

CloudStreams Connections	All properties under Connection configuration and Connection management are supported in the properties file.
	
	Connection property structure:
	
	`cloudstreamsconnection.<<package name>>.<<namespace>>.connectionconfiguration.<<property name>>=<<property value>>`
	
	Example:
	`cloudstreamsconnection.AssetsPackage.saleforceConnections\:sfdc_conn1.connectionconfiguration.cn..connectTimeout=180000`
	
	Supported properties:
	
	■ cn.providerUrl
	■ cn.connectTimeout
	■ cn.readTimeout
	■ cn.socketStaleCheck
	■ cn.validateAfterInactivity
	■ cn.retryCount
	■ cn.retryIfRequestSentOk
	■ cn.tcpNoDelay
	■ cn.socketLinger
	■ cn.sockBuffSize
	■ cn.reuseAddr
	■ cn.webproxyAlias
	■ cn.truststoreAlias
	■ cn.hostnameVerifier
	■ cn.keepAliveInterval
	■ cn.idleTimeout
	■ cn.enableCompression
	■ cr.username
	■ cr.password
	■ cr.authSchemeType

Asset	Property
	■ cr.preemptiveAuthEnabled ■ cr.securityRealm ■ cr.securityRealm ■ cr.clientKeyAlias ■ pr.httpContentCharSet ■ pr.protocolVersion ■ pr.userAgent ■ pr.useExpectCont ■ pr.useChunking ■ pr.followServerRedirects ■ pr.serverRedirectMax ■ rh.requestHeaderNames ■ rh.requestHeaderValues ■ oauth.consumerId ■ oauth.consumerSecret ■ oauth.accessToken ■ oauth_v10a.accessTokenSecret ■ oauth_v20.instanceURL ■ oauth_v20.refreshAccessToken ■ oauth_v20.refreshToken ■ oauth_v20.accessTokenRefreshURL ■ oauth_v20.accessTokenRefreshURLRequest ■ oauth_v20.accessTokenRefreshCustomESBService ■ oauth_v20.authorizationHeaderPrefix ■ oa.alias_key ■ aws.accessKey ■ aws.secretKey ■ aws.region

Asset	Property
	■ aws.signingAlgorithm See the *webMethods Deployer User's Guide* for description of these properties.

OAuth token configuration	The OAuth token configuration based on the OAuth version is available in the properties file.
	
	OAuth configuration property structure:
	
	`cloudstreamsoauthtokens.<<oath token alias>>.<<property>>=<<property value>>`
	
	Example: `cloudstreamsoauthtokens.testOAUTH1.accessToken`
	
	`={AES}LZx9vaF8gghTDI8YtM7Nxg\=\=`
	
	Supported properties:
	
	■ accessToken
	■ accessTokenSecret
	■ clientId
	■ clientSecret
	■ instanceUrl
	■ refreshToken
	■ refreshUrl
	
	See the *webMethods Deployer User's Guide* for description of these properties.

Large Data configuration	Large data configuration property structure:
	
	`cloudstreamslargefileconfig.LargeFileConfig.<<property name>>=<<property value>>`
	
	Example:
	`cloudstreamslargefileconfig.LargeFileConfig.wst.largedata.binaryStreamEnabled`
	
	`=true`
	
	Supported properties:
	
	■ wst.largedata.binaryStreamEnabled
	■ wst.largedata.threshold
	
	See the *webMethods Deployer User's Guide* for description of these properties.

Asset	Property

Database configuration	Database configuration property structure:
	`cloudstreamsdatabaseconfig.DatabaseConfig.<<property name>>=<<property value>>`
	Example:
	`cloudstreamsdatabaseconfig.DatabaseConfig.pg.jdbc.active=false`
	Supported properties:
	■ pg.PgMenConfiguration.perfDataEnabled
	■ pg.PgMenConfiguration.reportInterval
	■ pg.jdbc.active
	■ pg.publish.events
	See the *webMethods Deployer User's Guide* for description of these properties.

9 Deploying Assets using webMethods Deployer

Overview

You can deploy CloudStreams user-created assets such as connections, connector services and administrative assets or configurations such as oauth token, provider, subscriber configurations and database settings that reside on source webMethods repositories to target webMethods runtime components (runtimes) using webMethods Deployer.

Note:
You can deploy CloudStreams user-created assets in repository-based deployment only.

Building Assets

To support repository-based deployment, you must build the CloudStreams assets from sources available in a development environment or VCS and store the assets on a repository using the webMethods Asset Build Environment (ABE).

To build the assets

1. Go to *Software AG_directory*/common/AssetBuildEnvironment/master_build and open the `build.properties` file.

2. Configure the build.source.dir property if you want to build package composites along with other administrative assets or the build.source.project.dir if you want to build package composites separate from other administrative assets.\

   Note:
   To include CloudStreams administrative assets in the composite for deployment, you must manually copy or check-in the `Integration Server_directory`\instances\`instance_name`\packages\WmCloudStreams\config folder to the source directory.

3. Set the enable.build.IS property to `true`.

4. Ensure that you have assigned correct values to the sag.install.dir and build.output.dir properties.

5. Save the `build.properties` file.

6. Go to the `Software AG_directory`/common/AssetBuildEnvironment/master_build directory and run the `ant build` command. The build Ant target uses the properties in the `build.properties` file, validates the assets and dependencies, and generates the .acdl and data for the assets.

Deploying assets to other servers

For deploying CloudStreams assets to other Integration Servers, see the *webMethods Deployer User's Guide*.

10 CloudStreams Analytics

Overview

CloudStreams can publish key performance indicator (KPI) metrics and run-time events to the CloudStreams Analytics dashboard. The dashboard enables you to view and analyze the KPI metrics and run-time events of your CloudStreams providers, cloud connector services, and consumers.

The CloudStreams Analytics dashboard is powered by the Software AG MashZone NextGen Server. The dashboard tabs are backed by data feeds that communicate with the database using the JDBC connection pool of Integration Server to publish data at regular intervals.

Note:
The data feed of Software AG MashZone NextGen dashboard fetches data through a REST service hosted on Integration Server to retrieve metrics and events from the database and send it back to Software AG MashZone NextGen.

Note:
The CloudStreams Analytics dashboard has been updated and migrated from the legacy Software AG MashZone to Software AG MashZone NextGen.

The dashboard displays the following analytic views:

View	Can be filtered by...
Performance and availability for providers	
API usage by provider(s), cloud connector service(s), and consumer(s)	Provider and time period
Performance trends for provider(s) and cloud connector service(s)	Provider and time period
Error trends for provider(s) and cloud connector service(s)	Provider and time period
Invocation details for cloud connector service(s)	Provider, cloud connector service, and time period

The types of KPI metrics and run-time events in the views are listed below.

The Run-Time Events

The types of run-time events that can be published are as follows:

Event Type	Description
Lifecycle	A Lifecyle event occurs each time a CloudStreams provider is started or shut down.
Transaction	A Transaction event occurs each time a cloud connector service is invoked (successfully or unsuccessfully).

Event Type	Description
Error	An Error event occurs each time an invocation of a cloud connector service results in an error.
Policy Violation	A Policy Violation event occurs each time an invocation of a cloud connector service violates a policy that was set for its associated virtual service.
Monitoring	CloudStreams publishes key performance indicator (KPI) metrics, such as the average response time, fault count, and availability of all providers and connector services.

The Key Performance Indicator (KPI) Metrics

The types of KPI metrics that can be published are as follows:

Metric	Description
Availability	Indicates whether the provider was available to the specified consumers in the current interval. For information about intervals, see "Setting the Database Options for Publishing Performance Metrics and Events" on page 32. A value of 100 indicates that the connector or service was always available. If invocations fail due to policy violations, this parameter could still be as high as 100.
Average Response Time	The average amount of time it took the cloud connector service to complete all invocations in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller.
Fault Count	The number of failed invocations for a cloud connector service in the current interval.
Success Count	The number of successful invocations for a cloud connector service in the current interval.
Total Request Count	The total number of requests made by a connector, cloud connector service, or consumer in the current interval.

To enable CloudStreams to publish KPI metrics and events to the CloudStreams Analytics dashboard, you must set the options in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

Note:
The metrics do not include metrics for failed invocations unless you set the `pg.PgMetricsFormatter.includeFaults` parameter to true in *`IntegrationServer_directory`*`\`

`instances\instance_name\packages\WmCloudStreams\config\resources\wst-config.properties`.
For more information, see "Advanced Settings" on page 281.

Installation and Configuration Tasks

Before you can use CloudStreams Analytics, you must perform the following tasks:

1. "Creating the Schema and Tables Required by CloudStreams Analytics" on page 250
2. "Creating a JDBC Connection Pool for CloudStreams" on page 253
3. "Setting the Publishing Options for Performance Metrics and Events" on page 258
4. "Installing and Starting MashZone NextGen Server" on page 256
5. "Copying the Resource Files for CloudStreams Analytics" on page 256
6. "Configuring the Data Feed" on page 257
7. "Configuring the record limit for Transaction and Event tables" on page 257
8. "Importing the CloudStreams Analytics Dashboard" on page 257

Creating the Schema and Tables Required by CloudStreams Analytics

To create the schema and tables

1. Start the Database Component Configurator GUI as follows:

System	Action
Windows	On the **Start** menu, go to **Program > Software AG > Tools > Database Component Configurator**.
UNIX	Go to *Software AG_directory* and run the command `dbConfigurator.sh`.

2. In the **Action** area, in the **Type** list, click **create**.

   The **create** action type creates the database components you select in the **Action** area, and lets you create a database user and storage.

3. Click **Component**, and in the text box next to **Component**, select **CloudStreamsEvents**.

4. In the **Version** field, select **Latest**.

   The configurator will create the latest version of the database component.

5. In the **Connection** area, in the **RDBMS** field, select the RDBMS in which to create the database component.

   A URL specific to the selected database appears in the **URL** field.

6. Modify the URL to point to the database where the CloudStreams events should be collected.

   Sample URL formats for the DataDirect Connect JDBC 5.0 driver are displayed. Below is additional information for completing this field.

   - For Oracle, if you are going to create storage and the Data Purge database component, you must specify the sysLoginRole connection option on the URL (for example: `;sysLoginRole=sysdba`).
   - For DB2, if you are going to create database components in a schema other than the default schema for the specified database user, you must specify these connection options in the URL, where `AlternateID` is the name of the default schema used to qualify unqualified database objects in dynamically prepared SQL statements:

     `;AlternateId=`schema`;"InitializationString=(SET CURRENT PATH=`current_path,schema`)"`
   - For information about options supported by the DataDirect Connect JDBC 5.0 driver used by webMethods products, see *DataDirect Connect for JDBC User's Guide and Reference 5.0* in the *`Software AG_directory`*`/_documentation` directory.

7. In the **User ID** and **Password** fields, your entries depend on the task you are going to perform as follows:

If you are...	Specify...
Creating a database user and storage in Oracle or SQL Server	The database user and password to create. **Note:** For SQL Server, the user will be created and a default schema named dbo will be assigned to that user. **Note:** For Oracle, do not use the SYSTEM user to create the database components in the SYSTEM schema.
Creating a database user and storage in DB2 for LUW	The OS user to which to grant permissions, and the password for that user.

8. Select the **Create Tablespaces** check box. (To create tablespaces, you must have Admin privileges for the database.)

   Note:
   This check box is labeled **Create Database and Database User** for SQL Server, and **Create Tablespaces and Grant Permissions to OS User** for DB2.

Note:
For DB2 on Linux systems, tablespaces are created for each DB2 database. If you are creating webMethods database components in more than one DB2 database, either the tablespace directory or the tablespace names must be unique for each DB2 database.

9. In the **Admin ID** field, identify the database user or operating system user that has database administrator credentials to create the database user and storage. Supply the password for the user in the **Admin Password** field.

10. The next field and your entry depend on your RDBMS.

RDBMS	Field and Entry
Oracle or DB2 for LUW	In the **Tablespace Directory** field, identify the directory in which to create the tablespaces. For Oracle, use this field only if the DB_CREATE_FILE_DEST parameter is not set for your Oracle instance.
SQL Server	In the **Database** field, specify the database to create.

11. The default tablespace names are as follows:

RDBMS	Data Storage	Index Storage	BLOB Storage
Oracle	WEBMDATA	WEBMINDX	WEBMDATA
SQL	Primary	Primary	Primary
DB2 for LUW	WEBMDATA	WEBINDX	WEBMDATA
DB2 for iSeries	Default	Default	Default

12. For Oracle or DB2 for LUW, you can select the **Use Custom Tablespace Names** check box and specify custom tablespace names in the fields. You must have Admin privileges to do this.

    - For Oracle, the custom tablespace names will replace the defaults WEBMDATA and WEBMINDX.
    - For DB2 for LUW, the custom tablespace names will replace the defaults WEBMDATA, WEBMINDX, and WEBMBLOB. You can also specify a custom name to use for the buffer pool for webMethods products (WEBMBUFF by default).

    Note:
    webMethods products support all tablespace configurations deployed by users.

13. Click **Execute**.

    The execution information is displayed on the **Results** tab and is written to the log file `dcc_yyyymmddHHMMss` in the *`Software AG_directory`*`\common\db\logs` directory. If the execution is successful, all required tables are available.

If you intend to run the configurator more than once, you can set the current field values as the defaults for subsequent runs by clicking **Save Settings as Default**. You can also export field values to .xml files by clicking **Export**, then later import the values from a file by clicking Import Configuration. In each case, the values for the two **Password** fields are not saved.

Creating a JDBC Connection Pool for CloudStreams

To create a JDBC connection pool for CloudStreams

1. Start Integration Server and open Integration Server Administrator.
2. Go to the **Settings > JDBC Pools** page.
3. On the **Settings > JDBC Pools** page, click **Create a new Pool Alias Definition** and complete the fields to point to the tables you created in "Creating the Schema and Tables Required by CloudStreams Analytics" on page 250, as follows.

Field	Entry
Alias Name	Specify a name for the connection pool.
Alias Description	Description for the pool.
Associated Driver Alias	Database driver to use.
Database URL	URL for the database server. Sample URL formats for the DataDirect Connect JDBC 5.0 driver are displayed. Use the DataDirect Connect connection option `MaxPooledStatements=35` on the database URL. This connection option improves performance by caching prepared statements. For DB2, if Integration Server will connect to a schema other than the default schema for the specified database user, you must specify these connection options in the URL: `;AlternateId=`schema`;"InitializationString=(SET CURRENT PATH=`current_path,schema`)";MaxPooledStatements=35` `AlternateId` specifies the name of the default schema that is used to qualify unqualified database objects in dynamically prepared SQL statement.
User ID	Database user for Integration Server to use to communicate with the database.
Password	Password for the database user.

Field	Entry
Minimum Connections	Minimum number of connections the pool must keep open at all times. If you use this pool alias for more than one function, each pool instance keeps the specified number of connections open. For example, if you specify keeping at least 3 connections open, and the IS Core Audit Log and the Document History database components both use this pool, the pool keeps a total of 6 connections open: 3 for the IS Core Audit Log pool instance and 3 for the Document History pool instance. If your logging volume has sudden spikes, you can improve performance by making sure the connections needed to handle the increased volume open quickly. You can minimize connection startup time during spikes by setting this value higher, so that more connections remain open at all times.
Maximum Connections	Maximum number of connections the pool must keep open at all times. Calculate this value as part of the total possible number of connections that could be opened simultaneously by all functions and applications that write to the database. Make sure the total number does not exceed the database's connection limit. If one of the applications opens more connections than the database allows, the database will reject subsequent requests for connections from any application.
Idle Timeout	Period of time, in milliseconds, the pool can keep an unused connection open. After the specified period of time, the pool closes unused connections that are not needed to satisfy the **Minimum connections** value. **Note:** You should also consider setting *connection* timeouts as described in "SQL Statement Execution and Connection Timeouts" on page 255.

4. Click **Save Settings**.

5. Click **Return to JDBC Pool Definitions**.

6. In the **Functional Alias Definitions** table, locate the functional name **CloudStreams**.

7. Click **Edit** under the **Edit Association** column.

8. In the **JDBC Functional Alias** field under **Associated Pool Alias**, select the connection pool alias you created.

9. Make sure Integration Server can connect to the database by clicking **Test Connection**.

10. Click **Save Settings**.

11. Restart Integration Server.

SQL Statement Execution and Connection Timeouts

The CloudStreams event-sending component relies on the JDBC pools connection properties for configuring the DataDirect driver properties.

It is expected that any DataDirect driver settings will be configured via property settings in the connection URL. Minimally, it is recommended that timeouts be configured for connections and SQL statement execution since by default the DataDirect driver uses "0" for its default settings (which implies that no timeouts are used).

Following is a sample connection URL with the property settings `QueryTimeout` and `LoginTimeout`:

`jdbc:wm:oracle://<server>:<1521|port>;serviceName=<value>;QueryTimeout=<seconds>;LoginTimeout=<seconds>`

Note the following:

- The `LoginTimeout` property controls connection timeouts.

  If `LoginTimeout` is not specified, the connection pool will use the value of the global Integration Server property `watt.server.jdbc.loginTimeout` to attempt to connect to the database.

  The `watt.server.jdbc.loginTimeout` property is located in *`IntegrationServer_directory`*`\instances\`*`instance_name`*`\config\server.cnf`. This property sets the maximum time, in seconds, that the JDBC driver on Integration Server is to wait for a response while attempting to connect to a database. If Integration Server does not receive a response in the allotted time, it terminates the request, logs the error and moves on. The default value is 60 seconds.

  Note:
  The DataDirect property `LoginTimeout` takes precedence over the global Integration Server property `watt.server.jdbc.loginTimeout`, which means you can override the global property on a case-by-case basis.

- The `QueryTimeout` property controls timeouts for queries and for inserting events.

  Although the name `QueryTimeout` suggests its scope is limited to queries, it actually applies to *all* SQL statements, including INSERTs.

  If The `QueryTimeout` property is not specified, there is no timeout for SQL statements executing on connections in this pool. So it is recommended that you specify a `QueryTimeout` value for this pool.

If you experience timeouts while inserting events, you should consider increasing the `QueryTimeout` property (and/or reducing the CloudStreams property `pg.jdbc.batchSize`, which is described below).

- The CloudStreams property `pg.jdbc.batchSize` controls the size of the events batch.

  CloudStreams uses the optimized JDBC batch API to send its events to the relational database using prepared statements. The CloudStreams property `pg.jdbc.batchSize` controls how many events the sender can insert into a given batch (25 events by default). So, a batch of events is sent to the database every time the value of `pg.jdbc.batchSize` is reached. Increasing the batch size can improve database performance, but will also increase the statement execution time.

  The `pg.jdbc.batchSize` property is located in:

  IntegrationServer_directory`\instances\`*instance_name*`\packages\WmCloudStreams\config\resources\wst-config.properties`

  Additionally, if more than 3 seconds elapse since the last batch was sent, any received events will be sent to the database, even if the batch threshold has not been reached. This ensures there is not a substantial delay between the time an event is created and when it is inserted into the table.

Installing and Starting MashZone NextGen Server

To install MashZone NextGen and start the server

1. Install Software AG MashZone NextGen server from the Software AG Installer, as described in the *Installing Software AG Products* document.

   On the **Configure** page of the wizard, specify the license file, accept the default ports for the MashZone NextGen Server, and then continue.

2. Start the MashZone NextGen Server from the Programs menu.

Copying the Resource Files for CloudStreams Analytics

To copy the required resource files

1. Copy the file *SAG_HOME*`\CloudStreamsAnalytics\mashngapp\CloudStreams Analytics.zip` into the *MASHZONE_NG_HOME*`\prestocli\bin` folder, where *MASHZONE_NG_HOME* is the base directory of the MashZone NextGen installation.

2. Copy the contents (excluding the images folder) of the folder *SAG_HOME*`\CloudStreamsAnalytics\mashngapp\resources` into the *MASHZONE_NG_HOME*`\mashzone\data\resources` folder.

3. Copy the contents of the folder *SAG_HOME*`\CloudStreamsAnalytics\mashngapp\resources\images` into the *MASHZONE_NG_HOME*`\apache-tomcat\webapps\mashzone\images` folder.

Configuring the Data Feed

To configure the data feed

1. Open the `config.xml` file located in the *`MASHZONE_NG_HOME`*`\mashzone\data\resources` folder.
2. Set the host and port to point to the host and port of Integration Server where CloudStreams is installed.

   The default host and port are `localhost` and `5555` respectively.

Configuring the record limit for Transaction and Event tables

To configure the record limit for Transaction and Event tables

1. Open the `config.xml` file located in the *`MASHZONE_NG_HOME`*`\mashzone\data\resources` folder.
2. Set the value of the transactions and events fields to a positive integer value. The default record limit for transactions and events is 10000.

   Note:
   MashZone NextGen Server currently does not support pagination, so there is no way to limit the transaction and event records displayed at a time. The manual configuration of transaction and event records has been put in place in order to prevent the UI from becoming unresponsive when a substantially large number of transaction or event records are persisted in the DB.

Importing the CloudStreams Analytics Dashboard

To import the CloudStreams Analytics Dashboard

1. Open a command window under `MASHZONE_NG_HOME\prestocli\bin`, and run the `padmin importDashboard` command to import the CloudStreams Analytics dashboard into the MashZone NextGen Server.

   Note:
   The MashZone NextGen Server must be running for the `padmin importDashboard` command to run successfully. Further, specify `CloudStreamsAnalytics.zip` as the value for the `-f` or `--inputFile` parameter.
2. Navigate and login to the MashZone NextGen home page (http://<host>:<port>/mashzone/hub/home/index.html). The default port is 8080. The default credentials are Username: Administrator and Password: manage. Verify that the CloudStreams Analytics dashboard is listed as a dashboard.

3. Click the CloudStreams Analytics dashboard link to view the CloudStreams Analytics dashboard.

Setting the Publishing Options for Performance Metrics and Events

CloudStreams uses the Integration Server's JDBC connection pool to publish the performance metrics and events. To enable CloudStreams to publish performance metrics and events to the CloudStreams Analytics dashboard, you must set the options described in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

Using the CloudStreams Analytics Dashboard

After you have completed the "Installation and Configuration Tasks" on page 250, login to the dashboard as follows.

To login to the dashboard

1. Login to the MashZone NextGen home page.
2. Click the CloudStreams Analytics link to view the CloudStreams Analytics dashboard.
3. The dashboard's first view, called Performance Summary appears.

   The dashboard provides the following views:

   - "Performance Summary" on page 258
   - "API Usage" on page 259
   - "Performance Trends" on page 260
   - "Error Trends" on page 262
   - "Transaction" on page 263
   - "Event" on page 264

Performance Summary

This view provides a summary of the performance and availability of all CloudStreams providers. It lists all the providers and their availability, performance, and status.

To use the Performance Summary view

1. Open the CloudStreams Analytics dashboard, as described in "Using the CloudStreams Analytics Dashboard" on page 258.

By default, the Performance Summary is the first view displayed in the dashboard.

2. This view displays the following information about all CloudStreams providers installed and registered with CloudStreams:

Field	Description
Status	There are two possible statuses: ■ **OK**: The connector is operating normally. ■ **Error**: The reason for the error appears in the **Performance and Availability Summary** column.
Provider Name	The name of the provider, for example, WmSalesforceProvider.
Performance and Availability Summary	The reason for the provider's status. One reason for Error status would be Availability of less than 80%.
Performance	The average response time (in milliseconds) for a provider in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment it returns the response to the caller. For information about intervals, see "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.
Availability	The percentage of time that the provider was available during the current interval. A value of 100 indicates that the provider was always available. Availability of less than 80% results in Error status.

API Usage

This view shows the API usage of the providers, filtered by provider and time period.

To use the API Usage view

1. Open the CloudStreams Analytics dashboard, as described in "Using the CloudStreams Analytics Dashboard" on page 258.

2. Click the **Usage** tab in the dashboard.

3. Filter the metrics you want to view as follows:

Filter	Description
Cloud Provider	Choose a specific cloud provider or choose **All**.

Filter	Description
Period	The time period for which to view API usage: ■ **Last 90 Days**: The previous 90 days, including today. ■ **Last 30 Days**: The previous 30 days, including today. ■ **Last 7 Days**: The previous 7 days, including today. ■ **Last 1 Day**: The previous 24 hours. ■ **Last 1 Hour**: The previous hour.

4. This view displays the following information:

Field	Description
Providers by Number of Requests	Shows each provider's share (%) of requests received during the specified time period.
Providers by Data Volume	Shows each provider's share (%) of the request/response payload size sent and received during the time period.
Top 5 - Cloud Connector Services by Number of Requests	Shows the five cloud connector services that received the most requests during the specified period.
Top 5 - Cloud Connector Services by Data Volume	Shows the five cloud connector services that received/sent the largest amount of request/response payload data (in Kilobytes).
Top 5 - Consumers by Number of Requests	Shows the five consumers that sent the most requests.
Top 5 - Consumers by Data Volume	Shows the five consumers that sent the largest amount of request/response payloads (in Kilobytes).

Performance Trends

This view shows the following metrics, filtered by cloud provider and time period:

- Performance (average response time) of cloud connector services. Performance is shown by provider, consumer and individual cloud connector services.

  Average response time is the average time (in milliseconds) it took the cloud connector service to complete all invocations in the current interval. Response time is measured from the moment CloudStreams receives the request until the moment just before it returns the response to the consumer. By default, the average response time does not include metrics for failed invocations. For information about intervals, see "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

- Availability of cloud connector services. Availability is shown by consumer and individual connector services.

  Availability indicates the percentage of time that the cloud connector service was available during the current interval. A value of 100 indicates that the cloud connector service was always available. Only the time when the service is unavailable counts against this metric. If invocations fail due to policy violations, this parameter could still be as high as 100. For information about intervals, see "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.

To use the Cloud Connector Performance view

1. Open the CloudStreams Analytics dashboard, as described in "Using the CloudStreams Analytics Dashboard" on page 258.

2. Click the **Performance** tab in the dashboard.

3. Filter the metrics you want to view as follows:

Filter	Description
Provider	Choose a specific provider or choose **All**.
Period	The time period for which to view performance: ■ **Last 90 Days**: The previous 90 days, including today. ■ **Last 30 Days**: The previous 30 days, including today. ■ **Last 7 Days**: The previous 7 days, including today. ■ **Last 1 Day**: The previous 24 hours. ■ **Last 1 Hour**: The previous hour.

4. This view displays the following information:

Field	Description
Provider Performance Trends	Shows the average response time for each provider.
Top 5 - Cloud Connector Services by Average Response Time	Shows the average response time of the five fastest cloud connector services.
Slowest 5 - Cloud Connector Services by Average Response Time	Shows the average response time of the five slowest cloud connector services.

Error Trends

This view shows service request error trends.

To use the Cloud Connector Error view

1. Open the CloudStreams Analytics dashboard, as described in "Using the CloudStreams Analytics Dashboard" on page 258.

2. Click the **Error** tab in the dashboard.

3. Filter the metrics you want to view as follows:

Filter	Description
Provider	Choose a specific provider or choose **All**.
Period	The time period for which to view performance: ■ **Last 90 Days**: The previous 90 days, including today. ■ **Last 30 Days**: The previous 30 days, including today. ■ **Last 7 Days**: The previous 7 days, including today. ■ **Last 1 Day**: The previous 24 hours. ■ **Last 1 Hour**: The previous hour.

4. This view displays the following information:

Field	Description
Provider Error Trends	Shows the number of service request errors returned by each provider.
Top 5 - Services by Fault Count	Shows the five cloud connector services that returned the highest number of faults. The fault count indicates the number of failed requests for the service for the current interval. For information about intervals, see "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.
Lowest 5 - Services by Fault Count	Shows the five cloud connector services that returned the fewest number of faults.

Transaction

This view shows the run-time performance of cloud connector services. It can be filtered by provider, connector service, and time period.

To use the Transaction View

1. Open the CloudStreams Analytics dashboard, as described in "Using the CloudStreams Analytics Dashboard" on page 258.

2. Click the **Transaction** tab in the dashboard.

3. Filter the metrics you want to view as follows:

Filter	Description
Provider	Choose a specific provider or choose **All**.
Cloud Connector Service	Choose a specific cloud connector or choose **All**.
Period	Choose one of the following values or click **Timeframe All**. ■ **Last 90 Days**: The previous 90 days, including today. ■ **Last 30 Days**: The previous 30 days, including today. ■ **Last 7 Days**: The previous 7 days, including today. ■ **Last 1 Day**: The previous 24 hours. ■ **Last 1 Hour**: The previous hour.

4. This view displays the following information:

Field	Description
Provider	The provider name.
Cloud Connector Service	The cloud connector service name.
Connector Virtual Service	The connector virtual service.
Total Requests	Total number of requests for the service, sorted in descending order (by the services requested the most).

Field	Description
Fault Count	Number of faults that the service returned.
Success Count	Number of successful requests for the service.
Average Response Time	Average response time for the service.
Total Bytes of Data	Total size of all request/response payloads.

Event

This view shows the event results of all request transactions for cloud connector services. It can be filtered by provider, connector service, connector virtual service, event type, and time period.

To use the Event View

1. Open the CloudStreams Analytics dashboard, as described in "Using the CloudStreams Analytics Dashboard" on page 258.

2. Click the **Event** tab in the dashboard.

3. Filter the metrics you want to view as follows:

Filter	Description
Provider	Choose a provider or choose **All**.
Cloud Connector Service	Choose a cloud connector service or choose **All**.
Connector Virtual Service	Choose a connector virtual service or choose **All**.
Event Type	Choose any of the following event types or choose **All**. By default, the **Error Event** is selected. ■ **Error Event**: An Error event occurs each time a cloud connector service invocation results in an error. ■ **Policy Violation Event**: A Policy Violation event occurs each time a cloud connector service invocation violates a policy that was set for its associated virtual service. ■ **Transaction Event**: A Transaction event occurs each time a cloud connector service is invoked (successfully or unsuccessfully).

Filter	Description
	■ **Monitoring Event**: CloudStreams publishes key performance indicator (KPI) metrics, such as the average response time, fault count, and availability of all connectors and connector services. **Note:** To enable CloudStreams to publish these events, you must set the options described in "Setting the Database Options for Publishing Performance Metrics and Events" on page 32.
Period	Choose one of the following values or click **All**. ■ **Last 90 Days**: The previous 90 days, including today. ■ **Last 30 Days**: The previous 30 days, including today. ■ **Last 7 Days**: The previous 7 days, including today. ■ **Last 1 Day**: The previous 24 hours. ■ **Last 1 Hour**: The previous hour.

4. This view displays the following information:

Field	Description
Provider	The provider name.
Cloud Connector Service	The cloud connector service name.
Connector Virtual Service	The cloud connector virtual service name.
Event Type	The event type that resulted (Error, Policy Violation, Transaction or Monitoring).
Session ID	The SOAP invocation session ID that generated the event.
Target Name	The CloudStreams server target that generated the event.
Event Create Timestamp	The event timestamps, sorted in descending order (the most recent timestamps first).
Error/Alert Source	The source of errors and alerts generated by Error, Policy Violation and Monitoring events.
Error/Alert Description	Descriptions of the errors and alerts generated by Error, Policy Violation and Monitoring events.

Field	Description
Lifecycle Status	The status generated by a Lifecycle event.

A Admin Folder

Summary of Elements in this Folder

The following elements are available in this folder:

Element	Description
pub.cloudstreams.admin.connection:disableConnection	Disables a connection node.
pub.cloudstreams.admin.connection:enableConnection	Enables an existing connection node.
pub.cloudstreams.admin.connection:getConnectionStatistics	Returns current usage statistics for a connection node.
pub.cloudstreams.admin.connection:queryConnectionState	Returns the current connection state (enabled/disabled) and error status for a connection node.
pub.cloudstreams.admin.listener:disable	Disables an existing listener. Also, disconnects the listener.
pub.cloudstreams.admin.listener:enable	Enables an existing listener.
pub.cloudstreams.admin.listener:queryListenerState	Returns the current listener state (enabled/disabled) for a listener node.
pub.cloudstreams.admin.listener:listEnabledListeners	Returns a list of aliases for connector listeners in enabled state for a given connection alias.
pub.cloudstreams.admin.listener:update	Updates a CloudStreams connector listener node with a new set of inputs.
pub.cloudstreams.admin.service:update	Updates a CloudStreams connector service node from one version of connector API to a higher version.

Element	Description
pub.cloudstreams.admin.service:batchUpdate	Updates batches of CloudStreams connector service nodes from one version of connector API to a higher version.
pub.cloudstreams.upgrade:batchUpgrade	Updates batches of CloudStreams connector service nodes and connector listener nodes from a lower version of the connector API to a higher version.

pub.cloudstreams.admin.connection:disableConnection

Disables a connection node.

Input Parameters

connectionAlias **String**. Name of the connection node you want to disable.

Output Parameters

status **String**. Indicates whether the connection is disabled successfully. Values are:

- `true` - The connection is disabled.
- `false` - Disabling the connection failed.

statusMessage **String**. The status message of the error.

pub.cloudstreams.admin.connection:enableConnection

Enables an existing connection node.

Input Parameters

connectionAlias **String**. Name of the connection node you want to enable.

Output Parameters

status **String**. Indicates whether the connection is enabled successfully. Values are:

- `true` - The connection is enabled.
- `false` - Enabling the connection failed.

statusMessage **String**. The status message of the error.

pub.cloudstreams.admin.connection:getConnectionStatistics

Returns current usage statistics for a connection node.

Input Parameters

connectionAlias **String**. Name of the connection node for which you want usage statistics returned.

Output Parameters

connectionStatistics **Document List**. Information for each connection node. Keys are:

- *totalConnections*. **String**. Current number of connection instances.
- *busyConnections*. **String**. Number of connections currently in use by services, notifications, and listeners.
- *freeConnections*. **String**. Total number of connections created and available for use.
- *totalHits*. **String**. Number of times this connection node successfully provided connections since the last reset.
- *totalMisses*. **String**. Number of times this connection node unsuccessfully provided connections since the last reset (when the request timed out).

pub.cloudstreams.admin.connection:queryConnectionState

Returns the current connection state (enabled/disabled) and error status for a connection node.

Input Parameters

connectionAlias	**String**. Name of the connection node for which you want the connection state and error status returned.

Output parameters

response	**Document List**. Information about the connection state and error status. Keys are:
	■ *connectionState*. **String**. Current connection state (enabled/disabled).
	■ *hasError*. **Boolean**. Flag indicating if any error was detected on the connection. Values are:
	  ■ `true` - An error was detected.
	  ■ `false` - No error was detected.
	■ *lastErrorTime*. **String**. Indicates the last time an error occurred.

pub.cloudstreams.admin.listener:disable

Disables an existing listener. Also, disconnects the listener.

Input Parameters

listenerAlias	**String**. Name of the listener node you want to disable.

Output Parameters

status	**String**. Indicates whether the listener is disabled successfully. Values are: ■ `true` - The listener is disabled. ■ `false` - Disabling the listener failed.
statusMessage	**String**. The status message of the error.

pub.cloudstreams.admin.listener:enable

Enables an existing listener. When a listener is enabled, a connection is established with the streaming provider subscription endpoint, and the listener is then ready to listen.

Input Parameters

listenerAlias	**String**. Name of the listener node you want to enable.

Output Parameters

status	**String**. Indicates whether the listener is enabled successfully. Values are: ■ `true` - The listener is enabled. ■ `false` - Enabling the listener failed.
statusMessage	**String**. The status message of the error.
warning	**String List**. Contains a list of warnings related to the listener configuration.

pub.cloudstreams.admin.listener:queryListenerState

Returns the current listener state (enabled/disabled) for a listener node.

Input Parameters

listenerAlias	**String**. Name of the listener node for which you want the status to be returned.

Output parameters

listenerState	**String**. Current listener state (enabled/disabled). Values are: ■ `enabled` - The listener is enabled. ■ `disabled` - The listener is disabled.

pub.cloudstreams.admin.listener:listEnabledListeners

Returns a list of aliases for connector listeners in enabled state for a given connection alias.

Input Parameters

connectionAlias	**String**. Name of the connection node.

Output Parameters

listenerAlias	**String List**. A list of names of the listener nodes in enabled state, which depend on the given connection.

pub.cloudstreams.admin.listener:update

This service allows you to update a CloudStreams connector listener node with a new set of inputs.

Note:

- Make a backup copy of the existing package containing the CloudStreams listener service node before running the update service.

Input Parameters

listenerAlias	**String**. Name of the existing listener node which needs to be updated.
connectionAlias	**String**. Name of the new connection node to be referenced by the connector listener.
connectionGroupProperties	**Document**. Optional. Set of listener connection group properties.
subscriber	**Document**. Subscriber data of the listener.
event	**Document**. Optional. Event data of the listener.

Output Parameters

status	**String**. Indicates whether the listener is updated successfully. Values are: - `true` - The listener is updated. - `false` - Updating the listener failed.
statusMessage	**String**. The status message of the error.

pub.cloudstreams.admin.service:update

This service allows you to update a CloudStreams connector service node from one version of connector API to a higher version for various attributes such as the *Connection Alias* and the *Virtual Service Name*. If the connector service node was created with an earlier version of CloudStreams,

you must first migrate the service node before using the update service. For information on how to migrate a connector service node, see the *Upgrading Software AG Products* document.

Note:

- Make a backup copy of the existing package containing the CloudStreams connector service node before running the update service.
- If the update service is used to update a CloudStreams connector service node to a higher version of API and if backward compatibility is not supported by the SaaS provider, the CloudStreams connector service node may not function properly.
- It is recommended to use the update service to update a CloudStreams connector service node from a lower version of connector API to a higher version. Using the update service to update a CloudStreams connector service node from a higher version of API to a lower version is not supported.
- Updating a non-CUSTOM CloudStreams connector service node to CUSTOM type, that is, having an operation with either `input.body.transformer` or `output.body.transformer` property is also not supported.

By default, the update service runs under simulation mode, that is, no actual changes are made to the corresponding CloudStreams connector service node and only a check is performed if the service node can be updated successfully.

Input Parameters

cloudServiceName	**String**. Required. Name of the CloudStreams connector service node which needs to be updated.
connectionAlias	**String**. Optional. Name of the new connection alias. The CloudStreams connector service node will get updated to use this new connections alias. The connector associated with the new connection alias must be from the same provider package and should be of the same type, that is, REST/SOAP.
virtualServiceName	**String**. Optional. Name of the new virtual service name. The CloudStreams connector service node will get updated to use this new virtual service. For more information on virtual services, see “Virtual Services” on page 55.
simulate	**String**. Optional. Allowed values are true or false. Instructs the update service to run in simulation mode during which no changes are made to a CloudStreams connector service node, and only a check is performed if a service node can be updated successfully or not. Default value is true. Set simulate to false to persist the changes made to a CloudStreams connector service node.
force	**String**. Optional. Allowed values are true or false. Instructs the update service to continue updating a CloudStreams connector service node, and ignore errors it encounters related to missing headers, parameters, and custom input/output document types that can prevent the service node from being updated in a normal

scenario. Default value is false. Using force as true may result in an unusable CloudStreams connector service node with incorrect mappings and subsequent service execution failure.

Output parameters

success **Boolean String**. Required.

- `true` – If the CloudStreams connector service node was updated successfully.
- `false` – If the CloudStreams connector service node update has failed.

errorMessage **String**. Optional. Information on the cause of the error, if the update has failed.

pub.cloudstreams.admin.service:batchUpdate

This service updates batches of CloudStreams connector service nodes from a lower version of the connector API to a higher version.

Note:

- Make a backup copy of the existing package(s) containing the CloudStreams connector service nodes before running the batchUpdate service.
- If the batchUpdate service is used to update CloudStreams connector service nodes to a higher version of API and if backward compatibility is not supported by the SaaS provider, the CloudStreams connector service nodes may not function properly.
- It is recommended to use the batchUpdate service to update CloudStreams connector service nodes from a lower version of the connector API to a higher version. Using the batchUpdate service to update CloudStreams connector service nodes from a higher version of API to a lower version is not supported.
- Updating non-CUSTOM CloudStreams connector service nodes to CUSTOM type, that is, having an operation with either `input.body.transformer` or `output.body.transformer` property is also not supported.

If the connector service nodes were created with an earlier version of CloudStreams, you must first migrate the service nodes before using the batchUpdate service. For information on how to migrate a connector service node, see the *Upgrading Software AG Products* document.

The batchUpdate service may take a long time to complete if an input package contains large number of connections. This is because packages are reloaded at the end of the batchUpdate service, and a package with many connections takes more time to reload. For better performance, disable the connections before executing the batchUpdate service.

By default, the batchUpdate service runs under simulation mode, that is, no actual changes are made to the corresponding CloudStreams connector service nodes, and only a check is performed if the service nodes can be updated successfully.

Input Parameters

input

Document List. Required. Represents an array of input records to the batchUpdate service. At least one input record is required. Each input record has the following parameters:

- *packageName.* **String**. Required. Name of the package containing the CloudStreams connector service nodes to be updated.
- *nsFilter.* **Document**. Optional. A namespace filter in the form of text expression(s), which can be applied to selectively filter CloudStreams service nodes within the package specified by the `packageName` parameter. The namespace filter has the following parameters:

  – *patternStyle.* **String**. Optional. Allowed values are glob or regex. The style of include or exclude text expression(s). glob allows for UNIX glob like expressions that are used by UNIX shell commands, whereas regex allows regular expressions to be used (as used by Java and Perl languages). The default style is glob.

  – *include.* **String List**. Optional. An array of text expressions to include CloudStreams connector service nodes, which are to be updated. The default is to include all CloudStreams connector service nodes matched by the `oldConnectionAlias` parameter.

  – *exclude.* **String List**. Optional. An array of text expressions to exclude CloudStreams connector service nodes from being updated. The default is to exclude none.
- *oldConnectionAlias.* **String**. Required. Name of the old connection alias. All CloudStreams connector service nodes matched by the namespace filter and having this connection alias will be picked for an update.
- *newConnectionAlias.* **String**. Required. Name of the new connection alias. The matched CloudStreams connector service nodes will get updated to use this new connection alias. The connectors associated with the old connection alias and the new connection alias must belong to the same provider package and should be of the same type, that is, REST/SOAP.

simulate

String. Optional. Allowed values are true or false. Instructs the batchUpdate service to run in simulation mode during which no changes are made to CloudStreams connector service nodes, and only a check is performed if a service node can be updated successfully or not. Default value is true. Set simulate to false to

persist the changes made to a CloudStreams connector service node.

force **String**. Optional. Allowed values are true or false. Instructs the batchUpdate service to continue updating CloudStreams connector service nodes, and ignore errors it encounters related to missing headers, parameters, and custom input/output document types that can prevent the service nodes from being updated in a normal scenario. Default value is false. Using force as true may result in an unusable CloudStreams connector service node with incorrect mappings and subsequent service execution failure.

Output parameters

output **Document List**. Required. Represents an array of output records of the batchUpdate service corresponding to each input record of the batchUpdate service. Each output record has the following parameters:

- *packageName.* **String**. Required. Name of the package containing CloudStreams connector service nodes provided as an input to the batchUpdate service.
- *status.* **String**. Required. Allowed values are true or false. The overall status of the batchUpdate service; true if all input CloudStreams connector service nodes could be updated successfully, false otherwise.
- *statusMessage.* **String**. Required. A detailed error message in case the batchUpdate service could not run properly, or a summary message displaying how many CloudStreams connector service nodes have updated successfully and how many service nodes have failed to update.
- *node.* **Document**. Optional. An array of node records for each individual CloudStreams connector service node, which was acted upon by the service. Each node record has the following parameters:

  – *nsName.* **String**. Required. The fully qualified NS Name of the CloudStreams connector service node.

  – *success.* **String**. Required. Allowed values are true or false. The status of the CloudStreams connector service node; true if updated successfully, false otherwise.

  – *message.* **String**. Optional. A summary message if the CloudStreams connector service node was updated successfully, else an error message explaining why the service node could not be updated.

pub.cloudstreams.upgrade:batchUpgrade

This service updates batches of CloudStreams connector service nodes and connector listener nodes from a lower version of the connector API to a higher version.

Note:

- Make a backup copy of the existing package(s) containing the CloudStreams connector service nodes and connector listener nodes before running the batchUpgrade service.
- If the batchUpgrade service is used to update CloudStreams connector service nodes and connector listener nodes to a higher version of API and if backward compatibility is not supported by the SaaS provider, the CloudStreams connector service nodes and connector listener nodes may not function properly.
- It is recommended to use the batchUpgrade service to update CloudStreams connector service nodes and connector listener nodes from a lower version of the connector API to a higher version. Using the batchUpgrade service to update CloudStreams connector service nodes and connector listener nodes from a higher version of API to a lower version is not supported.
- Updating non-CUSTOM CloudStreams connector service nodes to CUSTOM type, that is, having an operation with either `input.body.transformer` or `output.body.transformer` property is also not supported.

If the connector service nodes and connector listener nodes were created with an earlier version of CloudStreams, you must first migrate the service nodes before using the batchUpgrade service. For information on how to migrate a connector service node and connector listener node, see the *Upgrading Software AG Products* document.

The batchUpgrade service may take a long time to complete if an input package contains large number of connections. This is because packages are reloaded at the end of the batchUpgrade service, and a package with many connections takes more time to reload. For better performance, disable the connections before executing the batchUpgrade service.

By default, the batchUpgrade service runs under simulation mode, that is, no actual changes are made to the corresponding CloudStreams connector service nodes and connector listener nodes, and only a check is performed if the service nodes can be updated successfully.

Input Parameters

input — **Document List**. Required. Represents an array of input records to the batchUpgrade service. At least one input record is required. Each input record has the following parameters:

- *packageName.* **String**. Required. Name of the package containing the CloudStreams connector service nodes and connector listener nodes to be updated.
- *nsFilter.* **Document**. Optional. A namespace filter in the form of text expression(s), which can be applied to selectively filter CloudStreams connector service nodes and connector listener

nodes within the package specified by the `packageName` parameter. The namespace filter has the following parameters:

– *patternStyle.* **String**. Optional. Allowed values are glob or regex. The style of include or exclude text expression(s). glob allows for UNIX glob like expressions that are used by UNIX shell commands, whereas regex allows regular expressions to be used (as used by Java and Perl languages). The default style is glob.

– *include.* **String List**. Optional. An array of text expressions to include CloudStreams connector service nodes and connector listener nodes, which are to be updated. The default is to include all CloudStreams connector service nodes and connector listener nodes matched by the `oldConnectionAlias` parameter.

– *exclude.* **String List**. Optional. An array of text expressions to exclude CloudStreams connector service nodes and connector listener nodes from being updated. The default is to exclude none.

- *oldConnectionAlias.* **String**. Required. Name of the old connection alias. All CloudStreams connector service nodes and connector listener nodes matched by the namespace filter and having this connection alias will be picked for an update.
- *newConnectionAlias.* **String**. Required. Name of the new connection alias. The matched CloudStreams connector service nodes and connector listener nodes will get updated to use this new connection alias. The connectors associated with the old connection alias and the new connection alias must belong to the same provider package and should be of the same type, that is, REST/SOAP.

simulate — **String**. Optional. Allowed values are true or false. Instructs the batchUpgrade service to run in simulation mode during which no changes are made to CloudStreams connector service nodes and connector listener nodes, and only a check is performed if a service node and a listener node can be updated successfully or not. Default value is true. Set simulate to false to persist the changes made to a CloudStreams connector service node and a connector listener node.

force — **String**. Optional. Allowed values are true or false. Instructs the batchUpgrade service to continue updating CloudStreams connector service nodes and connector listener nodes, and ignore errors it encounters related to missing headers, parameters, and custom input/output document types that can prevent the connector service nodes and listener nodes from being updated in a normal scenario. Default value is false. Using force as true may result in

an unusable CloudStreams connector service node with incorrect mappings and subsequent service execution failure or an unusable connector listener node with subsequent listener enablement or disablement failure.

Output parameters

output **Document List**. Required. Represents an array of output records of the batchUpgrade service corresponding to each input record of the batchUpgrade service. Each output record has the following parameters:

- *packageName.* **String**. Required. Name of the package containing CloudStreams connector service nodes and connector listener nodes provided as an input to the batchUpgrade service.
- *status.* **String**. Required. Allowed values are true or false. The overall status of the batchUpgrade service; true if all input CloudStreams connector service nodes and connector listener nodes could be updated successfully, false otherwise.
- *statusMessage.* **String**. Required. A detailed error message in case the batchUpgrade service could not run properly, or a summary message displaying how many CloudStreams connector service nodes and connector listener nodes have updated successfully and how many service nodes and listener nodes have failed to update.
- *node.* **Document**. Optional. An array of node records for each individual CloudStreams connector service node and connector listener node, which was acted upon by the service. Each node record has the following parameters:

  – *nsName.* **String**. Required. The fully qualified NS Name of the CloudStreams connector service node and connector listener node.

  – *success.* **String**. Required. Allowed values are true or false. The status of the CloudStreams connector service node and connector listener node; true if updated successfully, false otherwise.

  – *message.* **String**. Optional. A summary message if the CloudStreams connector service node and connector listener node was updated successfully, else an error message explaining why the service node and listener node could not be updated.

B Advanced Settings

Introduction

This appendix describes the parameters you can specify in the CloudStreams properties file:

Integration Server_directory\instances*instance_name*\packages\WmCloudStreams\config\resources\wst-config.properties

You can edit this file by using only a text editor. Before you edit the file, you should either:

- Shut down the Integration Server, make your changes and then restart the server,

  OR

- Make your changes and then reload the WmCloudStreams package.

Note:
Some parameters present in the `wst-config.properties` file can be edited from the CloudStreams Administration console in Integration Server Administrator. You do not have to restart Integration Server when you change parameters through the CloudStreams Administration console.

pg.backupFailedProxies

pg.backupFailedProxies
Indicates whether the backup services failed. The default is `false`.

pg.CollectionPool.

pg.CollectionPool.minThreads
Sets the minimum number of threads used for the Policy Engine. The default is `1`.

pg.CollectionPool.maxThreads
Sets the maximum number of threads used for the data collection work queue. The default is `8`.

pg.CollectionPool.forcefulShutdown
Specifies whether the data collection thread pool should shut down immediately or wait for queued tasks to complete during CloudStreams shutdown. The default is `false`.

pg.CollectionPool.poolName
Sets the name for the data collection thread pool. The default is `CollectionPool`.

pg.CollectionWorkQueue.

pg.CollectionWorkQueue.queueCapacity
Sets the capacity available for data collection tasks stored in the memory work queue. The default is `10000`.

pg.debug.

pg.debug.eventLoggerActive
This is an internal parameter. Do not modify.

pg.delayedRefresher.

CloudStreams cannot query for updates or receive deployed services until Integration Server is running. If Integration Server is not yet fully operational when CloudStreams starts, a delayed refresh helper is used to wait for Integration Server. This helper will periodically check on Integration Server's status.

pg.delayedRefresher.napMillis
Specifies the amount of time (in milliseconds) the delayed refresher helper waits before checking to see whether Integration Server is running. The default is `500`.

pg.ehcache.config.

pg.ehcache.config.file
The path to the ehcache configuration file. This path will be provided to Integration Server during the first startup, and subsequent startups will not use the local file; it will use the one under `IS/config/Caching`. Do not modify.

pg.email.

pg.email.charset
Specifies the character set to use for the subject line, email addresses, and message body of the emails when sending alerts or events. The default is `US-ASCII`.

pg.email.debug
This is an internal parameter. Do not modify.

pg.email.from
Specifies the email address used when sending events by email. The default is `targetName@IS-hostname`.

You can edit this parameter from the **From** field on the **CloudStreams > Administration > Email** page in Integration Server Administrator.

pg.email.resourceMimeType
Specifies the MIME type that CloudStreams uses to send the request and response payload attachments for transaction events that are sent by email. CloudStreams supports the following values:

- `application/gzip` (.gz)
- `application/zip` (.zip)
- `text/xml` (.xml)

The default is `application/gzip`.

pg.email.SenderActive
Specifies whether an email server is configured for CloudStreams. The default is `false`.

You can edit this parameter from the **SMTP Host Name/IP Address** field on the **CloudStreams > Administration > Email** page in Integration Server Administrator.

pg.email.smtpHost
Specifies the host name of the email server.

You can edit this parameter from the **SMTP Host Name/IP Address** field on the **CloudStreams > Administration > Email** page in Integration Server Administrator.

pg.email.smtpPort
Specifies the email port for the SMTP or SMTPS protocol. The default is `25`.

You can edit this parameter from the **Port** field on the **CloudStreams > Administration > Email** page in Integration Server Administrator.

pg.email.timeOut
Specifies the time out period (in milliseconds) when connecting to an e-mail server and sending e-mails. The default is `1000`.

pg.email.TLSEnabled
Specifies whether to use one-way transport-layer security (TLS). If set to true, the truststore configured for CloudStreams must include a certificate in the email server's certificate chain. The default is `false`.

You can edit this parameter from the **TLS Enabled** check box on the **CloudStreams > Administration > Email** page in Integration Server Administrator.

pg.email.userName
Specifies the user name of the email account used to log into the SMTP server.

You can edit this parameter from the **User** field on the **CloudStreams > Administration > Email** page in Integration Server Administrator.

pg.endpoint.

pg.endpoint.connectionTimeout
Specifies the time interval (in seconds) after which an HTTP connection attempt will timeout. Default: 30 seconds.

This is a global property that applies to the endpoints of all virtual services. If you prefer to specify a connection timeout for the endpoints of virtual services individually, set the **HTTP Connection Timeout** parameter in the virtual service's "Routing Protocols" processing step. This parameter takes precedence over `pg.endpoint.connectionTimeout`.

pg.endpoint.readTimeout
Specifies the time interval (in seconds) after which a socket read attempt will timeout. Default: 30 seconds.

This is a global property that applies to all virtual services. If you prefer to specify a read timeout for virtual services individually, set the **Read Timeout** parameter in the virtual service's "Routing Protocols" processing step. This parameter takes precedence over `pg.endpoint.readTimeout`.

pg.failedProxies.

pg.failedProxies.backupDir
The absolute or relative path to the `config` directory.

pg.http.

pg.http.ports
Specifies the HTTP ports on which CloudStreams is available.

You can edit this parameter from the **HTTP Ports Configuration** field on the **CloudStreams > Administration > General** page in Integration Server Administrator.

pg.https.ports
Specifies the HTTPS ports on which CloudStreams is available.

You can edit this parameter from the **HTTPS Ports Configuration** field on the **CloudStreams > Administration > General** page in Integration Server Administrator.

pg.IntervalPool.

The interval pool is used to schedule the processing of recurring tasks.

pg.IntervalPool.minThreads
Specifies the minimum thread count for this interval pool. The default is `1`.

pg.IntervalPool.maxThreads
Specifies the maximum thread count for this interval pool. The default is `1`.

pg.IntervalPool.forcefulShutdown
Specifies whether the interval thread pool should wait for queued tasks to complete during CloudStreams shutdown. Setting this parameter to true will cause CloudStreams to shut down immediately, without waiting for the tasks to finish. The default is `false`.

pg.IntervalPool.poolName
Specifies the name of the interval processor pool. The default is `IntervalPool`.

pg.jaxbFileStore.

pg.jaxbFileStore.consumerFileStore
Specifies the location of the locally persisted consumer applications that CloudStreams receives. This file is updated periodically. The default is `resources\consumers\consumers.xml`. The consumers are populated from the **CloudStreams > Administration > Consumers** page.

pg.jdbc.

pg.jdbc.active
Specifies whether to send events to CloudStreams JDBC tables (assuming no policies are violated).

You can edit this parameter from the **Database Publishing** check box on the **CloudStreams > Administration > Database options** page in Integration Server Administrator.

pg.jdbc.batchSize
Specifies the maximum number of records that the JDBC sender should send in a batch. The default is `25` records.

When the Policy Engine sends events to the database, it uses JDBC batch API for efficiency purposes. This parameter determines the size of the batch before the Datadirect driver sends records from the Integration Server JVM over to the database.

pg.keystore.

pg.keystore.keyStoreHandle
This is an internal parameter. You can modify the parameter from the **Solutions > CloudStreams > Administration > General** tab.

pg.keystore.trustStoreHandle
This is an internal parameter. You can modify the parameter from the **Solutions > CloudStreams > Administration > General** tab.

pg.lb.

pg.lb.http.url
Specify the primary HTTP load balancer URL and port number to use in `http://`hostname`:`portnumber format.

You can edit this parameter from the **Load Balancer URL (HTTP)** field on the **CloudStreams > Administration > General** page in Integration Server Administrator.

pg.lb.https.url
Specify the primary HTTPS load balancer URL and port number to use in `http://`hostname`:`portnumber format.

You can edit this parameter from the **Load Balancer URL (HTTPS)** field on the **CloudStreams > Administration > General** page in Integration Server Administrator.

pg.lb.failoverOnDowntimeErrorOnly
Controls CloudStreams's behavior for load-balanced endpoints. The default is `false`, which means that in a load-balanced routing scenario, if a service fault is encountered in the response coming from endpoint 1, then CloudStreams will immediately try the next configured endpoint. There is no distinction on the type of fault present in the response from endpoint 1. However, if this parameter is set to `true`, then CloudStreams will failover only if the service fault is a "downtime" error (that is, if it matches one of the strings defined in the file:

`*Integration Server_directory*\instances\*instance_name*\packages\WmCloudStreams\config\resources\downtime-patterns.txt`

pg.passman.

pg.passman.configFile
This is an internal parameter. Do not modify.

pg.PgMenConfiguration.

pg.PgMenConfiguration.perfDataEnabled
Controls whether CloudStreams sends run-time performance data to the relational database destination. The default is `true`.

> **Note:**
> If this parameter is disabled, CloudStreams will not collect performance metrics.

You can edit this parameter from the **Publish Performance Metrics** check box on the **CloudStreams > Administration > Database options** page in Integration Server Administrator.

pg.PgMenConfiguration.publishErrorEvents
Specifies whether to publish Error events to CloudStreams. The default is `false`.

You can edit this parameter from the **Publish Events (Error, Lifecycle, Policy Violation)** check box on the **CloudStreams > Administration > Database options** page in Integration Server Administrator.

pg.PgMenConfiguration.publishLifeCycleEvents
Specifies whether to publish Life Cycle events to CloudStreams. The default is `false`.

You can edit this parameter from the **Publish Events (Error, Lifecycle, Policy Violation)** check box on the **CloudStreams > Administration > Database options** page in Integration Server Administrator.

pg.PgMenConfiguration.publishPolicyViolationEvents
Specifies whether to publish Policy Violation events to CloudStreams. The default is `false`.

You can edit this parameter from the **Publish Events (Error, Lifecycle, Policy Violation)** check box on the **CloudStreams > Administration > Database options** page in Integration Server Administrator.

pg.PgMenConfiguration.reportInterval
Specifies the how often (in minutes) CloudStreams publishes performance data to CloudStreams. The default is `60` minutes.

You can edit this parameter from the **Publish Interval** field on the **CloudStreams > Administration > Database options** page in Integration Server Administrator.

pg.PgMenConfiguration.sieFlushInterval
Specifies the number of seconds before the accumulated invoked service events are pushed into the shared cache. The default is `2`.

pg.PgMenConfiguration.tickInterval
Specifies the amount of time (in seconds) between each interval processor iteration. This must be an evenly divisible fraction of the smallest policy interval, which is one minute. The default is `15`.

pg.PgMenSharedCacheManager.

pg.PgMenSharedCacheManager.lockTimeOut

Specifies the amount of time (in seconds) the shared cache waits to obtain a lock before timing out. The default is `5` seconds.

pg.PgMetricsFormatter.

pg.PgMetricsFormatter.includeFaults
Specifies whether service faults are included in the aggregated performance metrics. If set to true, the average, minimum, and maximum response times will include failed requests in the calculations. The default is `false`. For more information, "The Key Performance Indicator (KPI) Metrics" on page 249.

pg.policygateway.

pg.policygateway.targetName
Sets the name of the CloudStreams instance configured as a target. It is used to identify this instance of CloudStreams. The default is `your-target-name-here`.

You can edit this parameter from the **Target Name** field on the **CloudStreams > Administration > General** page in Integration Server Administrator.

pg.policygateway.repositoryLocation
This is an internal parameter. Do not modify.

pg.policygateway.deleteTempArtifacts
Specifies whether to delete artifacts that are temporarily persisted by the deployment receiver. The default is `true`.

pg.proxyLoader

pg.proxyLoader.proxyLocation
This is an internal parameter. Do not modify.

pg.rampartdeploymenthandler.

pg.rampartdeploymenthandler.signingCertAlias
Specifies the signing alias used to sign the outgoing response from CloudStreams to the original request service consumer.

You can edit this parameter from the **Alias (Signing)** field on the **CloudStreams > Administration > General** page in Integration Server Administrator.

pg.rampartdeploymenthandler.usernameTokenUser
This is an internal parameter. Do not modify.

pg.ReportingPool.

Reporting pool options affect outbound event publishing. The pool includes all events, including performance data, JDBC and SMTP.

pg.ReportingPool.minThreads

Specifies the minimum thread count for this reporting pool. Enabling more threads means that CloudStreams can send more events to the event destination faster. The default is `2`.

pg.ReportingPool.maxThreads
Specifies the maximum thread count for this reporting pool. Enabling more threads means that CloudStreams can send more events to the event destination faster. The default is `4`.

pg.ReportingPool.forcefulShutdown
Specifies whether the reporting pool should wait for queued tasks to complete during CloudStreams shutdown. Setting this parameter to true will cause CloudStreams to shut down immediately, without waiting for the tasks to finish. The default is `true`.

pg.ReportingPool.poolName
Specifies the name of the reporting pool. The default is `ReportingPool`.

pg.ReportingWorkQueue.

pg.ReportingWorkQueue.queueCapacity
Sets the capacity available for data collection tasks stored in the reporting work queue. Smaller in-memory collection and reporting queues reduce memory load on Integration Server, but can result in some messages being discarded. If you configure more memory for Integration Server, it can support more events in the queues. The default is `5000`.

pg.serviceReader.

pg.serviceReader.interval
This is an internal parameter. Do not modify.

wst.cloudConnectorService.

wst.cloudConnectorService.validation
Set this parameter to `true` if you want cloud connector services to be validated against the IS document types present in the service signature. The default is false.

wst.connfactory.

wst.connfactory.wirelogEnabled
Set this parameter to `true` if you want to capture additional debug messages that show the request's HTTP content that is sent over the wire to the native provider. The debug messages also show the response content that was received. This can be useful when debugging connector service invocations. These debug messages may contain account specific or user specific data, for example, email addresses, which may be considered as personal data or personal identifiable information. If for privacy concerns or to be in compliance with General Data Protection Regulation (GDPR), you want to delete the personal data of a user, you can manage the log files that contain the wire logging data from the file system. See the *webMethods Integration Server Administrator's Guide* for information on how to manage these logs, their related files, and specific identifiers to look for in the logs.

You can also edit this parameter from the **Connection Factory Wire Logging** option on the **CloudStreams > Administration > General** page in Integration Server Administrator.

Note:
Enabling wire logging will reveal sensitive data, for example, user credentials and authorization headers in the logs. It is recommended to filter out or mask the sensitive data before sharing the logs.

wst.default.tenant.

wst.default.tenant.id
This is an internal parameter. Do not modify.

wst.product.

wst.product.name
This is an internal parameter. Do not modify.

wst.largedata.

wst.largedata.binaryStreamEnabled
This property enables the large data handling capability. If enabled, it allows CloudStreams to send and receive large binary streams over HTTP/HTTPS. The default value of this property is `false`.

wst.largedata.threshold
This property specifies the large data threshold size in bytes. For service invocations, if the stream is greater than the threshold size specified here, the entire stream is not stored in memory. For this property to take effect, *wst.largedata.binaryStreamEnabled* must be enabled. The default value of this property is `5 MB`.