webMethods Dynamic Business Orchestrator Help

Version 10.3

October 2018

WEBMETHODS

This document applies to webMethods Dynamic Business Orchestrator Version 10.3 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: DBO-PDF-103-20181024

Table of Contents

About this Guide

This guide explains how to use webMethods Dynamic Business Orchestrator to create, control, and monitor business processes.

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". The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

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.asp" 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.

I About Dynamic Business Orchestrator

About Dynamic Business Orchestrator

webMethods Dynamic Business Orchestrator combines creating process models at design time with executing and monitoring the process instances at runtime. Dynamic Business Orchestrator includes the following features:

- Create process models in the Dynamic Process Development perspective in Software AG Designer based on a single clear description of the process model that is shared by the design time, runtime, and monitoring tools.
- Validate process models at design-time and runtime.
- Monitor process instances using the webMethods Business Console web user interface.
- Configure error handling for process instances.
- Execute process instances in a webMethods Integration Server cluster.

II Architecture and Components

Components

Dynamic Business Orchestrator

Dynamic Business Orchestrator is the run-time component that acts as the consumer of process model content.

Dynamic Business Orchestrator detects changes to the model content, depending on the mode in which you use the orchestrator:

- In development mode, the model content changes are detected using a private service in Designer.
- In production mode, a public service is used to scan the local repository and validate or load new models, as well as remove or clean up old models.

Dynamic Business Orchestrator provides public services for controlling and administering process models and keeps all installations in a cluster synchronized, depending on the model definition and configuration.

If a process model uses business rules or tasks, you must ensure that you have the following run-time components installed:

- webMethods Rules Engine
- webMethods Task Engine

For information about Rules Engine and Task Engine, see the documentation of the respective component.

Dynamic Business Orchestrator User Interfaces

Designer

Software AG Designer is the design time tool that enables you to create process model content. You can do the following design tasks using the Dynamic Process Development perspective in Designer:

- Create process models.
- Deploy a process model to the runtime for execution and upload the process metadata to the Process Audit database.

Business Console

The Processes tab of the webMethods Business Console user interface displays enables you to monitor, control, and analyze process instances and to view aggregated statistical information about all process instances.

When a process instance includes tasks or Agile Apps cases, you can view run-time details for the tasks or cases in Business Console.

Monitor

webMethods Monitor is a user interface in My webMethods that enables you to administer the process models .Monitor retrieves metadata for the process models from the Process Audit database and uses the metadata to perform administration tasks for a process model, for example, to enable or disable a process model for execution or tracking.

III Configuring Dynamic Business Orchestrator

Configuring Archive Settings

Dynamic Business Orchestrator emits information about every process instance , such as step details, errors, and logged fields to the process archive database. Over time, this database can grow large and start affecting the performance of both execution and querying. Archiving the process instance data helps to improve performance. Archiving is done using one of the following methods:

- Stored procedures
- Partitioning strategies

Stored procedures are built-in and can be scheduled to run regularly.

webMethods Dynamic Business Orchestrator uses stored procedures to archive audit data to the Archive database. When Dynamic Business Orchestrator executes a stored procedure to archive or delete audit data, the database performs the entire operation without further interaction with Dynamic Business Orchestrator.

When using stored procedures to archive, the audit data must be archived to the same database where the stored procedure is located.

You can set the following archive operations for audit data:

- archive - archives and moves the audit data to the Archive database, and removes it from the source tables.
- archive and delete - archives the audit data and deletes it from the source table, but does not move the data to another location.

Partitioning requires database administration expertise. Audit database partitioning can be done based on timestamps. For example, partitioning can be configured to automatically move data out of the main partition and into a separate time-bound partition. For certain database vendors this may require separate licenses.

After you archive or delete audit data, you can no longer view that data in My webMethods. However, you can still execute queries on the data in the Archive database using SQL statements.

If you use an Oracle database as an Archive database, you can define a recipient of email alerts when the Oracle Purge operation completes.

Archiving Data Using Partitions

In the default stored procedure method of archiving, the stored procedures search for the records to archive (or delete) row by row, based on the input criteria. This is generally not a problem for smaller databases, but the process can be time-consuming for a large database with many audit records to be archived.

As an alternative to using stored procedures to archive and delete Process Audit data, you can use database partitioning, an option that greatly decreases the time required

to archive and delete data. The database partitions themselves are a standard feature of each database vendor, although you may need to purchase a separate partitioning license from your database vendor if you do not already have one. Monitor provides Oracle, Microsoft SQL, and IBM DB2 database scripts to configure and manage your partitions.

Note: Partition archiving support is only provided for Process Audit data. You must continue to use stored procedures for all other audit data.

To archive or delete audit data with partitioning, the first step is to define the needed partitions. Then, when you archive a partition, the script moves it from your active Process Audit database to the archive Process Audit database, and operation that typically takes seconds to complete, compared with archiving by stored procedures, which can take hours. To delete data, you drop the relevant partition.

Each partition stores only those records that fall within the partition's date range based on the column, ATRESTTIMESTAMP. When creating partitions, adhere to the following rules:

- Create as many partitions as you need.
- Configure each partition with a non-overlapping date range.
- Define every Process Audit database table with identical partitions.

Monitor stores process instances that are still running in a partition named WM_FUTURE (Oracle and DB2) or partition 1 (MS SQL). As audit data is written to the Process Audit tables, Monitor automatically writes audit data to this partition. This partition stores all audit data that is not yet considered complete. When a process instance completes, Monitor updates the ATRESTTIMESTAMP with the final completion date and moves all associated audit entries to the appropriate partition. This guarantees that all related audit data for a process instance exists in the same partition.

Configuring Partitions

You can define as few or as many partitions as you require based on your data volume and archiving needs.

To create and manage partitions for Process Audit Log data, refer the readme.txt file for your database in the following directories:

- For Oracle: <*Software AG_directory*>\common\db\scripts\oracle\processaudit\75\ partition_support
- For IBM DB2: <*Software AG_directory*>\common\db\scripts\db2\processaudit\75\ partition_support
- For Microsoft SQL: <*Software AG_directory*>\common\db\scripts\mssql \processaudit\75\ partition_support

Configuring the Archive Database

To use non-partitioned archiving, you must define the Archive database.

The following instructions provide a high level overview of the steps for creating the Archive database. For complete instructions, see "Creating and Dropping Database Components" in *Installing Software AG Products*.

To configure data archiving

1. Using the Database Component Configurator, create the Archive database for the Process Audit schema.
   a. In the **Action** section, select the values listed in the following table:

Field	Properties
Action Type	Create
Action Component	Archive
Version	Latest

   b. In the **Connection** section, define the connection to your Archive database, as described in the following table.

Field	Properties
RDBMS	Select the database provider. The Process Audit and Archive databases must be of the same type.
URL	Database URL.
User ID	The name of the database user account. This must be a new user and have sufficient privileges to access both the source and target Process Audit database.
Password	The password for the database user account.

   c. In the **Create Database and Database User** section, define the database Administrator, as described in the following table.

Field	Properties
Admin ID	Add the Archive database administrator.
Admin Password	Password for the Archive database administrator.
Database	Name of the Archive database, for example, `wmProcessAuditArchive`.

d. Click **Execute**.

2. In the Database Administration console, assign to the database user the appropriate permissions for the tables in the Archive and Process Audit database.
3. Connect the Archive database to an Integration Server. For complete instructions on connecting Integration Server to a database, see the section on configuring databases in *webMethods Integration Server Administrator's Guide*.
4. Define a new JDBC connection pool alias settings.
   a. In Integration Server Administrator, click **Settings > JDBC Pools**.
   b. In Pool Alias Definitions, click **Edit**.
   c. Add the URL, user ID and password to match the Connection settings defined in the Database Component Configurator and click **Save Settings**.
5. Define the JDBC pools for the Archive database.
   a. In Integration Server Administrator, click **Settings > JDBC Pools**.
   b. In Functional Alias Definitions, click **Edit** for Archiving.
   c. In **Associated Pool Alias**, select the alias and click **Save Settings**.
   d. Click **Restart**.
6. Configure the default archiving parameter in the OPERATION_PARAMETER table.
   a. In Designer, run the pub.monitor.archive:setOperationParameters service. pub.monitor.archive:setOperationParameters sets the values you specify in the `OPERATION_PARAMETER` table of the Archive database.
   b. Specify the input parameters listed in the following table.

Parameter	Entry
PROCESS_SCHEMA	To archive data from the Process Audit Log tables, specify the following information for your database provider: ■ **Oracle**: Process Audit Log database user ■ **SQL Server**: Process Audit Log database name ■ **DB2**: Process Audit Log schema name

Parameter	Entry
ISCORE_SCHEMA	To archive data from the IS Core Audit Log database, specify the following: ■ **Oracle**: IS Core Audit Log database user ■ **SQL Server**: IS Core Audit Log database name ■ **DB2**: IS Core Audit Log schema name
DBO_PROCESSES	**Boolean** Optional. Determines whether Dynamic Business Orchestrator processes are archived. Valid values are: ■ **TRUE**. Dynamic Business Orchestrator processes are archived. ■ **FALSE**. Default. Process Engine processes are archived.
DBO_PROCESS_STATUS	**String** Optional. When the value of the DBO_PROCESSES parameter is set to **TRUE**, the value of this parameter determines the statuses of the process data that services use to archive the Dynamic Process Orchestrator process data. Dynamic Process Orchestrator process data in other statuses is retained in the IS Core Audit Log and Process Audit Log database components. Valid values are: ■ **102**. Default. Completed Dynamic Process Orchestrator processes. ■ **103**.Default. Cancelled Dynamic Process Orchestrator processes. ■ **104**. Default. Terminated Dynamic Process Orchestrator processes. **Note:** The criteria values of this parameter are separated by commas.

7. Set database permissions to give the Archive database user permissions to select and delete data from the IS Core Audit Log tables, the Process Audit Log tables, or both, depending on the data you want to archive. To do so, execute the following SQL statement:

   ```
   GRANT SELECT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE, INSERT
   ANY TABLE
   ```

   Verify that the database user has the required permissions for the Archive tables.

Using MySQL Community Edition 5.7 with BPM

You must perform the following configuration tasks to use MySQL Community Edition 5.7 for business process development and management:

- Configure the MySQL Community Edition 5.7 driver for use with Integration Server.
- Create database tables in the Database Component Configurator.
- Configure a JDBC connection pool alias in Integration Server.
- Configure the MySQL Community Edition 5.7 driver for process upload.

Configuring the MySQL Community Edition 5.7 Driver for Use with Integration Server

To use MySQL Community Edition 5.7 with Integration Server, you must configure the required database driver. You can configure the database driver after installing Integration Server.

To configure the database driver for MySQL Community Edition 5.7

1. Download the JDBC driver for MySQL Community Edition 5.7 and place it in the following directory:

   Software AG_directory\common\lib\ext

2. In a text editor, open the ini.cnf file from the following directory:

   Software AG_directory\IntegrationServer\instances\IS_*instance_name* \bin

3. Add the following entry to the application.classpath property:

   %COMMON_LIB_EXT%*mysql-connector-java.jar*

   where *mysql-connector-java.jar* is the jar file for the MySQL Community Edition 5.7 driver.

4. Save your changes to ini.cnf and close the file.

5. Restart Integration Server.

Configuring Database Tables for MySQL Community Edition 5.7

Use the following procedure to configure database tables for MySQL Community Edition 5.7 in the Database Component Configurator.

To configure database tables in the Database Component Configurator

1. In a text editor, open the setEnv.bat file from one of the following directories, depending on where you installed the Database Component Configurator:
   - *Software AG_directory*\IntegrationServer\common\db\bin, if you installed the Database Component Configurator together with Integration Server.
   - *Database Component Configurator_directory* \common\db\bin, if you installed the Database Component Configurator separately.
2. Add the following classpath entry for the MySQL Community Edition driver jar file to the setEnv.bat file:

   set CLASSPATH=%CLASSPATH%;%DCI_HOME%\..\lib\ext*mysql_driver_jar*

   Example: set CLASSPATH=%CLASSPATH%;%DCI_HOME%\..\lib\ext\mysql-connector-java.jar
3. Use the following URL to create database tables in the Database Component Configurator:

   `jdbc:mysql://<server>:<3306|port>/`*`database_name`*

   Example: `jdbc:mysql://mysqldomain.com:3306/sampleDB`

Configuring JDBC Connection Pool Aliases in Integration Server

Use the following URL to configure a JDBC connection pool alias for MySQL Community Edition 5.7 on the Settings > JDBC Pools screen in Integration Server Administrator:

```
jdbc:mysql://&lt;server&gt;:&lt;3306|port>/<databaseName>?
relaxAutoCommit=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone
=integration_server_timezone
```

You must specify connection options for the `relaxAutoCommit`, `useLegacyDatetimeCode`, and `serverTimezone` parameters. For example, you can provide the connection options as follows:

```
jdbc:mysql://<server>:<3306|port>/databaseName?
relaxAutoCommit=true&useLegacyDatetimeCode=false&serverTimezone=PST
```

For more information about how to create a connection pool alias in Integration Server, see *webMethods Integration Server Administrator's Guide*.

Configuring the MySQL Community Edition 5.7 Driver for Process Upload

To use MySQL Community Edition 5.7 for process development, you must configure the required database driver in order to upload processes to the database. You configure the database driver after installing the Software AG Designer Dynamic Process Development perspective.

Important: To use MySQL Community Edition 5.7, you must select the **Use Integration Server JDBC pool parameters** option on the Window > Preferences > Software AG > Process Development > Process Audit Database page in Designer.

To configure the database driver for process upload

1. Download the JDBC driver for MySQL Community Edition 5.7 and place it in the following directory:

   Software AG_directory\Designer\eclipse\plugins\com.webmethods.process.metadatadeployer_10.1.0.0000-xxxx\lib

2. In a text editor, open the plugin manifest file from the following directory:

   Software AG_directory\Designer\eclipse\plugins\com.webmethods.process.metadatadeployer_10.1.0.0000-xxxx\META-INF\manifest.mf

3. Append the driver jar classpath to Bundle-ClassPath.

4. Restart Designer.

Process Archive Tables

This section lists the database tables for which you have to set permissions before you run the data archive process. Make sure you have the permission to archive the tables specified for the Dynamic Business Orchestrator version you are using.

- PRA_PROCESS_ACTION
- WMCUSTOMFIELDDEFINITION
- PRA_STEP_LOOP_LOGGED_FIELD
- PRA_STEP_LOGGED_FIELD
- PRA_ERROR
- PRA_PROCESS
- PRA_PROCESS_CUSTOM

- PRA_PROCESS_AT_REST
- WMPROCESSDEFINITION
- WMPROCESSIMAGE
- PRA_PROCESS_RECENT
- PRA_PROCESS_STEP
- PRA_PROCESS_STEP_LOOP
- WMPROCESSTASK
- WMPROCESSTASKSTEP
- WMPROCESSTASKUSER
- PRA_STEP_TRANSITION
- PRA_STEP_MESSAGE
- WMSERVICE_MIN_MAX
- WMSTEPDEFINITION
- WMSTEPTRANSITION DEFINITION

Controlling Access to Monitor Resources

My webMethods Server administrators determine which Monitor pages in My webMethods a user can access by assigning access privileges. For example, you can configure My webMethods so that a user can view pages related to monitoring process instances, but not allow the user to view pages related to monitoring services.

My webMethods Server administrators also determine which Monitor actions a user can perform by assigning functional privileges. For example, you can allow a user to view documents, but not to resubmit documents.

A My webMethods Server administrator can assign access and functional privileges to a user, group, or role.

My webMethods Server administrators can also assign *data-level security* to audit data, such as business processes, services, or documents. The data-level security or privileges determine which type of audit data a user can manage. The administrator assigns these privileges to a role. For example, the Service Administrator role can be allowed to act on service audit data.

For more information about access management of Monitor pages and administrative functions in My webMethods, see *Working with My webMethods* and *Administering My webMethods Server*.

Granting Users Access to Monitor Pages

Only a My webMethods Server Administrator user can grant access privileges. In My webMethods, use the **Navigate > Applications > Administration > System-Wide > Permissions Management** page to assign access privileges.

The following table describes the access privileges you can assign for Monitor pages.

To allow users to	In the Access Privileges section, select the check box
View process models that are available for monitoring.	**Administration > Business > Business Processes**
Archive data from the IS Core Audit Log and Process Audit Log databases.	**Administration > Business > Data Management**
View data about process instances.	**Monitoring > Business > Process Instances**
View data about services.	**Monitoring > Integration > Services**
View data about documents.	**Monitoring > Integration > Documents**

Granting Users the Ability to Perform Monitor Actions

Only a My webMethods Server Administrator user can grant functional privileges. In My webMethods use the **Navigate > Applications > Administration > System-Wide > Permissions Management** page to assign functional privileges.

The following table describes the functional privileges you can assign for Monitor pages.

To allow users to	In the Functional Privileges section, select the check box
Archive data or archive and delete data from the IS Core Audit Log and Process Audit Log databases.	**Data Management > Archiving**
Perform actions on Dynamic Business Processes.	**Business Monitoring > Processes > DBO - All Dynamic Process Actions**

Identifying the Audit Data on Which Users Can Perform Actions

My webMethods Server administrators can limit the types of data that a user can view or manage. This type of access control is referred to as data-level security. If a user belongs to more than one role, that user has access to all of the types of data and functions granted to all of the roles of which that user is a member.

To limit access to audit data on a role basis, you must:

- Enable data security as described in "Enabling Data-Level Security" on page 28.
- Configure role access to available process audit data, as described in "Identifying Processes, Services, and/or Documents on Which a Role Can Act" on page 29.

How Data-Level Security Works with Functional Privileges

Functional privileges are global across all of the data to which a user has been granted access. For example, assume the following two conditions:

- The role `HR` is granted the functional privileges to start and stop process instances and is granted data-level security access to the `newHire` process. As a result, users assigned to the `HR` role can view, start, and stop instances of the `newHire` process.
- The role `Interns` is granted data-level security access to the `ProblemReporting` process. As a result, users assigned to the `Interns` role can view instances of the `ProblemReporting` process.

If a user is assigned to *both* the `HR` and the `Interns` roles, because functional privileges are global and the `HR` role has the privilege to start and stop processes, the user assigned to both roles is able to start and stop not only instances of the `newHire` process, but also instances of the `ProblemReporting` process.

If you want to limit privileges, one straight-forward way to do so is to set up two user accounts. For example, assume that you want to give a user the ability to start and stop instances of the `newHire` process, but you also want that user to be able to only view instances of the `ProblemReporting` process. For this scenario, you could set up user account `joeHR` and assign the user account `joeHR` to the `HR` role, and then set up user account `joeIntern` and assign the user account `joeIntern` to the `Interns` role. When logged in as `joeHR`, the user can view, start, and stop `newHire` process instances. When logged in as `joeIntern`, the user can only view `ProblemReporting` instances.

Note: Data-level security is currently only supported in a single server environment.

Enabling Data-Level Security

When data-level security is disabled, users have unrestricted data access and can access all audit data. If you want to limit the data to which users have access, enable data-level security and then specifically identify the data to which different user roles have access.

To enable data-level security for Monitor

1. In Integration Server Administrator for the Integration Server that hosts the WmMonitor package, click **Packages > Management**
2. Click the **Home** icon for the **WmMonitor** package.
3. Select the **Enable Data Level Security** check box.
4. In the **Data Level Security Administrator** field, type the user name of a user who has access to all My webMethods data and pages.
5. Click **Submit**.
6. Reload the **WmMonitor** package.

Identifying Processes, Services, and/or Documents on Which a Role Can Act

When data-level security is *disabled*, the users with access privileges can view the pages listed in the table as follows in the table below:

Pages that display data for	User can view
Services	Audit data for *all* services.
Documents	All logged documents.
Process instances	Audit data for *all* process instances.

When you *enable* data-level security, by default, roles are blocked from accessing information about any processes, services, or documents. After you enable data-level security, you must configure data-level security for specific roles to identify the processes, services, and/or documents that each role can view and act on. After you have configured data-level security for roles, if a user belongs to multiple roles, that user will be able to work with all of the processes, services, and documents identified in all the roles to which the user belongs.

To identify the data on which a user role can act

1. In My webMethods, click **Navigate > Applications > Administration > System-Wide > User Management > Roles**
2. Search for the role for which you want to configure data-level security, and edit the role details. For more detailed instructions, see *Administering My webMethods Server*.
3. To configure data-level security for processes:
   a. On the Edit Role page, click the **Data Level Security** tab, and then click the **Business Process** link. My webMethods displays the list of all processes the role can currently access. The list is empty if no processes have been added yet.

b. To allow a role to access processes, click **Add Processes**. On the Add Processes page, select the processes you want to allow this role to access, and click **OK**.

c. Click **Apply**.

4. To configure data-level security for services, repeat step 3 but click the **Service** link.

5. To configure data-level security for documents, repeat step 3 but click the **Document** link.

Configuring Central User Management

If you want My webMethods users to perform Monitor tasks using their My webMethods user name and password, you must enable and configure central user management. With central user management, when a My webMethods user issues a Monitor request, My webMethods Server invokes a service in the WmMonitor package on Integration Server to handle the request.

The service is invoked using the user name and password of the requesting user, and Integration Server authenticates the user. If the user name and password do not match an Integration Server user, Integration Server uses central user management to authenticate the user.

For complete information about enabling and configuring central user management, see the PDF publication *webMethods Integration Server Administrator's Guide*. Central user management may already be configured in your environment. If not, follow the instructions in *webMethods Integration Server Administrator's Guide* to enable and configure it.

Note: If you do not use central user management, you must ensure that each Monitor user defined in My webMethods has a corresponding user account defined in Integration Server.

Verifying the Configuration of Central User Management in Integration Server

To verify the configurations of central user management in Integration Server

1. In Integration Server Administrator for the Integration Server that hosts the WmMonitor package: **Security > User Management**

2. Verify that the **Central User Management** field is set to **Configured**. If it is not, ask the administrator for that Integration Server to configure central user management.

3. In Integration Server Administrator for the Integration Server that hosts the WmMonitor package: **Settings > Resources**

4. Under **Single Sign On with My webMethods Server**, verify that **MWS SAML Resolver URL** field is set to `https://mws-host:mws-port/services/SAML`. If it is not, ask the administrator for that Integration Server to configure single sign-on.
5. In Integration Server Administrator for the Integration Server that hosts the WmMonitor package: **Settings > Extended**
6. Click **Edit Extended Settings** and verify that the following key/value pair is included in the extended settings:

   ```
   watt.server.auth.samlResolver=http://mws-host:mws-port/services/SAML
   ```

   If the setting is not defined, ask the administrator for that Integration Server to configure the setting.

Adding My webMethods Users to the Monitor Access Control Lists

To add My webMethods users to the Monitor Access Control Lists (ACLs)

1. In Integration Server Administrator for the Integration Server that hosts the WmMonitor package: **Security > ACLs**
2. In the **Select ACL** field, click **MonitorAdministrators ACL**.
3. Click **Add** under the **Allowed** list to view the current groups in the Select Role/Group dialog box.
4. In the **Provider** field, click **Central**.
5. Type an asterisk (*) in the **Search** field and then click **Go** to populate the list of roles and groups.
6. Click **My webMethods Users** to add that role to the **Allowed** list.
7. In the **Select ACL** field, select **MonitorUsers ACL**.
8. Repeat steps 3 - 6 to add the **My webMethods Users** role to the MonitorUsers ACL.
9. Click **Save Changes**.

Customizing How Monitor Sets Up ACLs When Using Central User Management

By default, Monitor sets the ACLs for the WmMonitor services based on My webMethods functional privileges. This enables users to perform all actions for which they have functional privileges. However, you can configure Monitor so that it does not automatically set the ACLs; if you do so, *you must* set the ACLs for the WmMonitor services.

If a user has the functional privilege to perform an action in My webMethods and you fail to assign the corresponding ACLs to WmMonitor services, the user will receive errors in the My webMethods user interface.

To customize how Monitor sets up ACLs when using central user management

1. In the Integration Server Administrator for the Integration Server that hosts the WmMonitor package: **Packages > Management**
2. Click the **Home** icon for the WmMonitor package.
3. To enable Monitor to automatically set the ACLs based on My webMethods functional privileges, select the **Add 'My webMethods Users' role to 'MonitorUsers' ACL** check box. To prevent Monitor from doing so, clear the check box.
4. Click **Submit**.

Configuring Database Connection Retries

You can configure the number of times that Monitor attempts to connect to a database (such as the Process Audit Log database) from which it reads data. If Monitor cannot connect in the specified number of tries, it logs the error to the host Integration Server's error log.

To configure Monitor connection attempts

1. In Integration Server Administrator for the host Integration Server: **Packages > Management**.
2. Click the **Home** icon for the WmMonitor package.
3. In the **Database Retries** field, specify the number of tries.
4. Click **Submit**.

Identifying the My webMethods Server that Hosts the Monitor User Interface

The Integration Server that hosts the WmMonitor package must know which My webMethods Server hosts the Monitor user interface to enable the user interface and package to communicate.

To identify the My webMethods Server that hosts the Monitor user interface

1. In Integration Server Administrator for the host Integration Server: **Packages > Management**.
2. In the WmMonitor row, click the **Home** icon.
3. Set the first five fields in the **Configuration Settings**.

> **Note:** By default, the My webMethods Server port number is 8585. Enter a different port number in the **MWS Port** field only if a non-default port was

specified during installation of My webMethods Server. If no value is entered, the **MWS Port** value is set to 8585.

4. Change any of the remaining configuration fields as necessary.
5. Click **Submit**.

IV Using Command Central to Manage Dynamic Business Orchestrator

About Administering Dynamic Business Orchestrator

You use Command Central to manage Dynamic Business Orchestrator. The Dynamic Business Orchestrator run-time component is a layered product of Integration Server. You can also monitor run-time statuses for a Dynamic Business Orchestrator instance.

Dynamic Business Orchestrator Lifecycle Actions

You can start, stop, restart, pause, and resume a Dynamic Business Orchestrator instance from the Command Central web user interface. The following lifecycle operations are available:

- **Start**. Start a stopped Dynamic Business Orchestrator instance.
- **Stop**. Stop a running Dynamic Business Orchestrator instance.
- **Restart**. Restart a running Dynamic Business Orchestrator instance.
- **Pause**. Pauses the execution of all running process instances, but keeps the Dynamic Business Orchestrator instance available during maintenance tasks. Pausing the Dynamic Business Orchestrator instance does not affect the status of the running process instances.
- **Resume**. Resumes the execution of previously-paused process instances.

On the **Overview** page for an instance, you can view the run-time status of a Dynamic Business Orchestrator instance in the status column.

You can also use the `sagcc get monitoring runtimestatus` command in the Command Central command line interface to check the runtime status of the Dynamic Business Orchestrator instance.

Dynamic Business Orchestrator General Properties

To access the General Properties configuration for a Dynamic Business Orchestrator instance, click a Dynamic Business Orchestrator instance name and navigate to **Configuration** > **General Properties**. You can update general properties, such as **Maximum DBO Execution Threads** and **DBO Execution Threads Running**.

Use the following commands to update the general properties of a Dynamic Business Orchestrator instance with the Command Central command line interface:

- To view the existing general properties, use the `sagcc get configuration data` command.
- To update the configuration properties, use the `sagcc update configuration data` command.

For more information about the commands, see *Software AG Command Central Help*.

Dynamic Business Orchestrator License Management

For Dynamic Business Orchestrator , you can configure the license, view the details of the license that is configured, and retrieve the location of the license file using the Command Central web user interface. To access the License Keys for a Dynamic Business Orchestrator instance, click a Dynamic Business Orchestrator instance name and navigate to **Configuration** > **License Keys**.

Note: You cannot change the location of a Dynamic Business Orchestrator license file.

You can view the content and location of the license file, and update the license file of a Dynamic Business Orchestrator instance in the Command Central command line interface using the following commands:

- To view the license details and location of a Dynamic Business Orchestrator license, use the `sagcc get configuration data` command.
- To add a Dynamic Business Orchestrator license key file with the specified alias to the Command Central license key manager, use the `sagcc add license-tools keys` command.
- To update a license key file assigned to the specified license key alias, use the `sagcc update configuration license` command.

For more information about the commands, see *Software AG Command Central Help*.

Dynamic Business Orchestrator Logs

You can view, download, and search all Dynamic Business Orchestrator logs in the Command Central web user interface. To access the logs for a Dynamic Business Orchestrator instance, click a Dynamic Business Orchestrator instance name and then the **Logs** tab.

To list and retrieve the Dynamic Business Orchestrator logs in the Command Central command line interface use the following commands:

- To list the log files available for a Dynamic Business Orchestrator instance, use the `sagcc list diagnostics logs` command.
- To retrieve log entries from a log file, use the `sagcc get diagnostics logs` command.

For more information on command line commands, see the *Software AG Command Central Help*.

V Creating Process Models

Creating Dynamic Business Processes

With Dynamic Business Orchestrator, process models are created in the Dynamic Business Development perspective in Designer and they mostly follow the BPMN 2.0 specification. The following topics provide information specific to creating dynamic business processes:

- Model Versioning
- Enforced Use of Gateways
- Disaster Recovery and Exception Handling
- Process Validation

For other information about creating business process models, see *webMethods BPM Process Development Help*.

Model Versioning

webMethods Monitor invokes public run-time services to activate a particular model version and to retrieve the currently active model version. By default, model versions are *not active* for execution. The policy of a single active model version is enforced by the runtime.

Saving a New Process from an Existing Process Model

In the Dynamic Process Development perspective of Designer, you can save an existing process model as a new process. Use this operation to copy the process model to a different location with a different name and project, but keep any existing mappings. Another way to use this operation is to keep the project name and process name and save the process model with a different version, creating a new version of the same process.

To save a new process for a process model:

1. Select the process model in the **Solutions** view.
2. Right-click the process model and select **Save As**, or click **File** > **Save As**.
3. Specify the fields listed in the following table in the **Save As** window:

Field	Value
Project Name	From the drop-down menu with existing projects, select a project for the new process.

Field	Value
Process Name	Specify a name for the new process.
Version	Specify the version of the new process.

4. Click **OK** to save the new process.

Mapping Data in Dynamic Business Orchestrator

In Dynamic Business Orchestrator you can use data mapping and edit directly the maps for a selected process step. Process step data mapping is done in the **Pipeline view** and the **Flow view** and all mapping information is saved with the model in the local workspace. You can add a transformer in the pipeline view when mapping. The services used as a transformer change the pipeline value of inputs. You can edit the input pipeline data of a step, which results in the pipeline data transforming before reaching the step.

Different steps support different types of mapping. For example:

- A Start Message Event Step supports only output data mapping.
- An End Message Event Step supports only input data mapping.
- A Service Task Step supports both input and output data mapping.
- A Gateway Step does not support any data mapping. Gateway steps only carry data without editing it.

To edit data mapping

1. In Designer, open the Dynamic Process Development perspective and the required process model.
2. Select the process step for which you want to edit the data mapping and go to the **Properties** view.
3. Click **Inputs/Outputs**.
4. Click **Edit Data Mapping**, or right-click the step and select **Edit Input Data Mapping** or **Edit Output Data Mapping**.
5. Edit the data mapping using the **Pipeline view** and the **Flow view**.

   Note: Each step has a left and right side available for mapping and these sides represent the input data maps and the output data maps respectively.

6. Click **Save**.

Editing Data in the Flow View

The Flow view is a part of the Dynamic Process Development Perspective in Designer. The Flow view provides the ability to create complex flow logic for the input and output of each process step (BPM step) that supports mapping. Flow elements (also called flow steps) are added to the Flow view by dragging the flow elements from the palette on the side and dropping them onto the Flow view. Using any element of the primary flow step types allows you to control the data in a flow.

You can edit the input or output flow logic of a process step by selecting the left/input side or the right/output side of the step respectively. When the left side (input side) of the process step is selected, then the  icon is displayed above the left side of the step, and the title of the Flow view shows **<*process step name*> INPUT**. When the right side (output side) of the process step is selected, then the  icon is displayed above the right side of the step, and the title of the Flow view shows **<*process step name*> OUTPUT**.

The concepts for both input and output mapping are the same and the following explanation is also valid for output data mapping.

In Designer a process step can perform work, for example via a service task. The service task can do an action such as retrieve data from a database, or print a status message. The service that performs that work usually requires input data (also known as the service's input parameters). In addition to the service's input parameters is the input data of the process step itself. The process step normally has input data that arrives into the process step from the upstream pipeline data and that upstream pipeline data is available to be mapped into the service's input parameters. The input mapping done in the Flow view and the Pipeline view is used to tie the upstream pipeline data to the service's input parameters.

The flow steps in the Flow view are executed, which enables you to affect the pipeline data both before it arrives as input to the service task as well as after it is returned as output from the service task.

In the Flow view, for process steps that support mapping, there is a single default flow **MAP**. Clicking on that **MAP** element in the Flow view results in the Pipeline view containing the following components:

1. The process step's upstream pipeline data under the column heading **Pipeline In**.
2. The service's input parameters under the column heading **<*process step name*> In (<*process step type*>)**.

You can add connections from the upstream pipeline data to the service's input parameters.

When you have multiple flow elements in the Flow view, unless these elements are explicitly mapped in a previous flow element, the service's input parameters are only shown in the Pipeline view for the last flow element. This occurs because only the last flow element has the service task as a downstream element. To ensure the parameters are displayed properly, make sure to map your data gradually as you add further flow

elements into the Flow view. Using this approach, the service's input parameters will be available for each successively-added flow element.

Tip: Disabling subsequent elements in the Flow view can change whether the service's input parameters are shown for previous flow elements. After the mapping is done and the subsequent flow elements are reenabled, all mapped data will remain.

For more information on flow services and the flow view, see the topics on building services and mapping data in flow services in the *webMethods Service Development Help*.

Running a Process Step's Flow Elements

You can execute a process step's flow logic independently from running the whole process. Executing just the process step's input flow or output flow enables you to verify that the flow logic for each process step is producing the expected results.

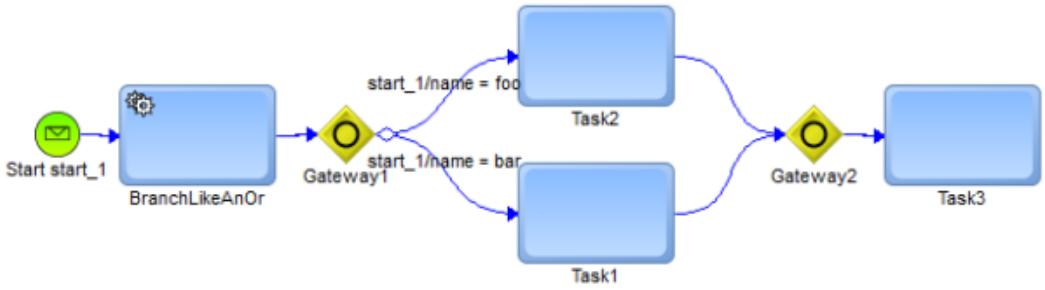
To run a process step's flow elements:

1. Select the input or output (left or right) side of a process step in the Designer Dynamic Process Development Perspective.
2. Open the Flow view.
3. In the flow view, right-click and select **Run As** > **Dynamic Flow**.
4. Provide values for the required input fields.
5. Click **OK**.

   The **Results** view is opened and the output for the process step's flow logic is displayed.

Using Gateways to Branch and Merge Processes

With Dynamic Business Orchestrator, when branching or merging processes, you must use gateways, for example:



In this example, Gateway 1 introduces two run-time rules, "start_1/name=foo" and "start_1/name=bar", which enforce that at least one live sequence flow will get executed.

If the sequence flow fails, the process model will fail at Gateway 2, instead of Task 3. The gateways help troubleshoot which flow causes the failure and re-adjust the process model accordingly.

Important: All non-gateway steps can only have a single input/output unconditional sequence flow.

Using Boundary Timer Events

You can add a boundary timer intermediate event to all activity types except for user and manual activities. Based on the activity type, boundary events are:

- Interrupting - when the boundary event interrupts the step activity. When the timer expires, the step stops and the process follows only the transition(s) from the boundary event.
- Non-interrupting - when the boundary event does not interrupt the step activity. When the timer expires, the process follows the transition(s) from the boundary event as well as the transition(s) from the activity.

A boundary event can have one or more output transitions. The output transitions from a boundary event do not support transition conditions.

When Dynamic Business Orchestrator receives the first input for a step with a event, it creates a timer object in memory. If Dynamic Business Orchestrator receives all step inputs before the timer expires, Dynamic Business Orchestrator cancels the timer.

When a server starts running the service of a step, Dynamic Business Orchestrator creates a timer object in memory. If the server finishes running the service before the timer expires, Dynamic Business Orchestrator cancels the timer.

If the timer expires, the step has timed out and transitions to the timeout transition defined for the step. The step produces a process transition document that identifies the next step to run. Dynamic Business Orchestrator publishes the document and the triggers for the specific target step, model, and model version.

Defining Timer Conditions for Boundary Events

To define a timer condition

1. In the process editor, click a boundary event to select it.
2. In the Properties view, click **Timer Condition**.
3. In the **Timer Condition** page, from the **Source** list, select the source of the timer value:
   - **Static Value**- specify a fixed period of time for the timeout value in the format days, hours, minutes, seconds, and milliseconds.
   - **Field Value**- define the timeout value dynamically by specifying the name of a data field present in the pipeline from an upstream document. You can specify

both top-level and nested fields. The value is in milliseconds. For example, if the field value is 60000, the timeout value will be 60000 milliseconds (one minute).

Note: The set of listed Field Values includes *all* process data items. You need to select a data item that will be present in the pipeline when the process executes. That requirement is a normal part of process design. The added support of Case steps provides an enhancement of a global data model, which leads directly to *all* process data items being presented for the Timer Condition.

4. Save the process.

Creating a Start Message Event from an Integration Server Document

You can create a start message event on the process canvas by dragging and dropping an Integration Server document from the Integration Server package navigator.

Note: You can initiate process instances that use Digital Event Services events that are defined in Integration Server by dragging and dropping an Integration Server document with the IS_DES_CONNECTION alias to the process canvas.

Using Event Subprocess

Event subprocesses are used within another process or subprocess and can act as a replacement for process-wide error handlers and timeout handlers. Event subprocesses have a dashed line border.

No inbound or outbound sequence flows are allowed from and to an event subprocess.

Event subprocesses is triggered with a single of the following start events:

- Start Error
- Start Message
- Start Timer

Interrupting a Process Using Event Subprocess

An event subprocess can either interrupt the main process that contains it, or run simultaneously. When the event subprocess is triggered using a Start Error event, the event subprocess always interrupts the main process. When the event subprocess is triggered using a Start Message or Start Timer event, the event subprocess does not interrupt the main process and runs simultaneously by default.

To configure the event subprocess to interrupt the main process:

1. Edit your process model in the Dynamic Process Development perspective in Software AG Designer.
2. Select the Start Timer or Start Message event in the event subprocess.
3. Go to **Properties** > **General** and select the **Interrupting** checkbox.
4. Save your changes.

Using Complex Gateway and Join Expressions

You build a join expression in Designer. Use the following procedure to specify the step's incoming transitions and place them into a logical statement. During runtime, the join is satisfied when the conditions in the expression are true.

If no join expression is defined, process validation will fail with an error stating that the join condition is invalid.

To define a complex join expression

1. Open a process and click a complex gateway.
2. In the Properties view, click the **Join Expression** page.
3. Use the editor controls to create the expression you want.

   Note that you can only select expression transitions and terms from a pre-defined list. Typing or pasting custom terms into the expression field is not supported.
4. Save the process.

Working with Process Steps

You create steps on the process editor's canvas by dragging a step type from the Palette view to the canvas and connecting the steps with transitions to create a process. Steps are categorized by what they do, specified in their properties, and also by their function in the process.

Step Labels

Software AG Designer enables you to apply a label to each step in your process model. It is possible for step labels to be empty, and for the same label value to be used more than once in the same process model. This enables models to be more accurately imported from other modeling tools in XPDL format.

You can add, remove, or modify a label value at any time by clicking the step to select it, and then editing the step **Label** field on the **General** page in the Properties view.

Task, call activity, and subprocess steps are always created with a default label, for example, “Task1,” with subsequent steps numbered incrementally. Event and gateway steps can be created with or without a label. When applied, the label uses the same format, for example, Gateway1, Message Event1.

Note: If you delete a step with a default name, that name is not reused in the process model. For example, if you add steps named Task1 and Task2 and then delete Task2, the next step is named Task3.

You can set the following preferences for a label:

- **Default step label location** determines the default placement of task, call activity, and collapsed subprocess step labels (on the step or below the step). In addition, you can set the position of the step labels in a process with the **Position label on step/below step** button  on the tool bar.
- **Automatically update step names when adding documents/services via drag and drop** determines whether step labels change after a drag and drop action.
- **Show event and gateway labels by default** determines whether or not a label is created when an event or gateway step is added to the process.

Step Inputs and Outputs

Each step in a process has information that flows into and out of it. Information flowing into a step is called *input*, and information flowing out of a step is called *output*. A process itself can also have inputs and outputs, such as when calling a process from a call activity step.

Process data assigned in Designer to flow in and out of steps needs to be mapped to *physical data* that the underlying services require in order for the process to execute.

Step inputs and outputs are used to define flow signatures, branching and looping logic in the process and data logging for examination at run time.

Step inputs and outputs are used to generate the signatures for the generated services that implement the process. If the underlying implementation of the step requires different physical data than this process data, the data must be mapped in the generated flows.

Process data follows a pipeline model, where all data that is input to a step must be output upstream in the process from that step.

Data can therefore enter the process in two ways:

- In a receive step, a subscription document can trigger or join the process, and output data for that step and into the pipeline
- In an activity step, the step can output new process data into the pipeline

While you can add new inputs to any step, the process will not be valid (for example, ready to be built) until all step inputs are first selected as outputs of an upstream receive or activity step.

Designer can automatically map inputs and outputs in the following circumstances:

- Step A is linked to step B, and the output of step A has the same name as the input of step B
- An activity step input name is the same as the document or service input name
- A Receive Task output document is the same type as its incoming document type

In all but these cases, you must manually map step input and output data.

Show and Hide Inputs and Outputs

You can configure whether to show or hide step inputs and outputs by default in the Preferences window, and you can also toggle the show/hide behavior using a button on the tool bar.

To show and hide step inputs and outputs

1. To set the default behavior for showing inputs and outputs, click **Window > Preferences > Software AG > Process Development > Appearance**, and select or clear the **Show inputs and outputs on steps by default** check box.
2. To show or hide step inputs and outputs in the open process, select or clear the **Show: Inputs/Outputs** check box on the process editor's toolbar.

Create Inputs and Outputs

Inputs and outputs are created in the same way, but they have different requirements due to their roles in a process. Outputs from steps create pipeline data, and are available to select as inputs to steps that are downstream in the pipeline. Inputs to all steps must exist upstream in the pipeline. If this is not the case, the issue is reported in the Problems view.

To edit the data mapping of fields inside a document or service, select **Edit Data Mapping** on the Inputs / Outputs page of the Properties view of the document or service whose data you want to map. Alternatively, right-click the step and select **Edit Data Mapping** from the context menu.

For more information about data mapping in Designer, see the *webMethods Service Development Help*.

To create an input or output

1. Select a step in the process editor.
2. On the Inputs / Outputs page in the Properties view, click **Create new input** in the Inputs section or **Create new output** in the Outputs section.

Important: All inputs must exist upstream in the pipeline. If you create a new input that does not yet exist upstream, you must create an output to feed the new input before completing the process.

3. Create a new input or output, or select an input from upstream in the pipeline:
   - If you create a new input or output, configure the **Name**, **Type**, and **Description**, and select the **List** check box if the input is an array. If you select a **Document Reference**, select the document from the Choose Document window.

     Tip: When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.

   - If you create an input based on an existing output from upstream in the pipeline, you do not need to configure the **Name**, **Type**, or **Description**. Designer populates the values automatically when you select the existing output.

     Important: Unnamed inputs and outputs are not saved.

     Text entered in the **Description** field is included in the HTML Documentation Report.

Remove Inputs and Outputs

To remove a step input or output

1. Select a step in the process editor.
2. On the Inputs / Outputs page in the Properties view, click **Remove input from step** in the Inputs section or **Remove output from step** in the Outputs section.

Important: If the inputs or outputs of a step are changed such that they do not match what is displayed in the process editor, the process will not refresh its inputs and outputs automatically. You must refresh them by editing the inputs or outputs on the step's Inputs / Outputs page in the Properties view. Remove the old inputs or outputs from the step, and use the **Auto-populate based on service signature** button to assign the new inputs or outputs.

Auto-Populate Inputs and Outputs

You can automatically populate the inputs and outputs of a step from its underlying IS service, Web service, task, or rule. This underlying information is known as the *service signature*.

Auto-populating step inputs and outputs allows Designer to do the data mapping of the step inputs and outputs. If you do not auto-populate with the service signature, you must manually map the data to the appropriate service. Click the **Edit Data Mapping** link

on the **Inputs/Outputs** page in the Properties view, or right-click a step and click **Edit Data Mapping**.

Note: Most steps have a single **Edit Data Mapping** right-click menu option. Call activity steps and task steps have two mapping options in their context menus: **Edit Input Data Mapping** and **Edit Output Data Mapping**. Empty steps do not have data to map, so they have no data mapping capability.

To auto-populate a step input or output

1. Select a step in the process editor.
2. On the **Inputs / Outputs** page in the Properties view, click **Auto-populate inputs based on service signature** in the Inputs section or **Auto-populate outputs based on service signature** in the Outputs section.

   Text entered in the **Description** field is included in the HTML Documentation Report.

Log Inputs and Outputs

In the Dynamic Process Development perspective and in the Process Debugging perspective, you can select fields from input and output documents for logging. You can also create aliases for the logged fields, which makes locating them in webMethods Monitor easier.

Input and output field logging is part of the Dynamic Business Orchestrator audit logging mechanism.

Logged fields can be viewed on the Process Instance Detail page in webMethods Monitor.

Note: Before you can select input and output document fields for logging, you must first define step inputs and outputs.

To select a step input or output document field for logging

1. Select a step in the process editor for which you have defined inputs and outputs.
2. On the **Logged Fields** page in the Properties view, click **Expand** to expand the **Inputs** and **Outputs** trees to display the fields available in the documents.
3. Select the check boxes that correspond to the document fields you want to log.
4. If you want to define an alias for a document field, type an **Alias** name.

   The alias defaults to the name and path of the selected field, but it can be modified to any alias for viewing in webMethods Monitor.

   Note: You can create the same alias for more than one field on a step, but this is not recommended, as it will make monitoring the fields at run time difficult.

Input and Output Types

The following step input / output types are available when configuring a step input / output. To select a list, choose the Input / Output type and then select the **List** check box.

Input / Output Type	Description
Boolean	True or false.
Boolean list	A one-dimensional boolean array.
Byte	Signed integer. The value must be greater than or equal to -128 but less than or equal to 127.
Byte list	A one-dimensional byte array.
Char	A single unicode character.
Char list	A one-dimensional character array.
Date	Date and time.
Date list	A one-dimensional date array.
Double	Double-precision floating point number.
Double list	A one-dimensional double array.
Float	Standard-precision floating point number.
Float list	A one-dimensional float array.
Integer	Signed integer. The value must be greater than or equal to -2147483647 but less than or equal to 2147483647.
Integer list	A one-dimensional integer array.
Long	Signed integer. The value must be greater than or equal to -9223372036854775808 but less than or equal to 9223372036854775807.

Input / Output Type	Description
Long list	A one-dimensional long array.
Short	Signed integer. The value must be greater than or equal to -32768 but less than or equal to 32767.
Short list	A one-dimensional short array.
Object	A data type that does not fall into any of the data types described in this table, and is not declared to be one of the basic Java classes supported natively by Integration Server.
Object list	A one-dimensional object array.
Document Reference	Select an Integration Server document type in the Choose Document window. The document you select becomes the input or output type. **Tip:** When an Integration Server connection is required but not available, Designer prompts you to connect. If no Integration Server is configured, Designer prompts you to configure one so you can connect to it.
String	A string of characters.
String list	A one-dimensional string array.

Defining a Global Process Specification

When you configure a call activity step to invoke a BPMN callable process, you also define a global process specification in the process you call. This includes inputs to and outputs from the callable process, allowing you to access process data.

The inputs of a callable process can be any available pipeline process data from previous steps in the process. Similarly, the outputs can be any process data you choose to include. The call activity step passes the entire pipeline to a start none event in the callable process, and the callable process automatically returns its resulting pipeline to the parent.

Designer automatically uses the defined global process specification (inputs and outputs) to populate the call activity step inputs and outputs. This happens when you drag and drop the child process onto the process editor canvas, or when you select the process on the **Implementation** page in the Properties view of the call activity step.

Tip: Click the Auto-populate button in both sections on the Inputs / Outputs page in the Properties view of the call activity step to update the inputs and outputs of the call activity to match the defined global process specification.

To define a global process specification

1. In the process editor, open the process you want to configure as a callable process.
2. Click anywhere in the design canvas to select the process.
3. On the **Global Process Specification** page in the Properties view, specify the documents that should be used to invoke the callable process in the **Input Specification for Callable Process** section. For more information, see "Create Inputs and Outputs" on page 48 and "Remove Inputs and Outputs" on page 49.
4. In the **Output Specification for Callable Process** section specify the documents you want returned to the call activity that called the process (you may need to scroll to the right to see this section). For more information, see "Create Inputs and Outputs" on page 48 and "Remove Inputs and Outputs" on page 49.
5. Save the process.

Adding an AgileApps Case Type to a Business Process in Designer

To add a case type to a process in Designer

1. In the Dynamic Process Development perspective, open a process in the process editor and add a **Case Task** as a step.
2. Click **Properties > Implementation**.
3. On the Implementation tab, click **Browse**.
4. In the Cases dialog box, select an AgileApps application and then a case type for the application.

   Designer populates the **Inputs/Outputs** for the step with the case object fields. You map the input and output fields in the same way that you perform mapping in a regular DBO process.

When you run the process, a case instance is automatically created in AgileApps. You can then modify the case business data in AgileApps or assign the case to a case agent.

Basic Process Properties

Properties Page	Property	Description
General	**Process Display Name**	The process display name of the process model. Can be modified at any time and supports non-ASCII characters.
	Process ID	A system-generated process identifier. Not editable. The **Process ID** consists of the process project name and the process name, separated by a slash (/). Process ID only supports ASCII characters.
	Version	The current version of the process. The initial process version is 1.
	Created By	The user name of the creator of the process. Not editable.
	Description	Your descriptive information about the process, for documentation purposes only. Text in process descriptions is searchable.
Documentation	**Documentation Fields**	Local and default documentation fields to document in the process. Documentation fields are searchable.
	Documentation Field Value	Value for the assigned **Documentation Field**.
Stages	**Stages**	A list of the stages created within this process model.
	Add Stage	Click to add a stage.
	Delete Stage	Click to delete a selected stage.

Properties Page	Property	Description
	Stage Details	Configuration information about the selected stage.
	Name	Name of the stage (read-only).
	Description	Optional. Description of the stage.
	Start Milestone	Selected start milestone for the stage.
	End Milestone	Selected end milestone for the stage.
	Condition	Specified condition to define a stage breach.
	Stop Tracking On Breach	Stops stage processing for all remaining stages in the process instance when a stage breach occurs in this stage, and only one stage breached EDA event is emitted. Remaining stages are not tracked and will be shown as Incomplete in Monitor. The check box is cleared by default.
Global Process Specification		Use this page to define the inputs and outputs for a callable process. A global process specification, along with a start none event, is used when a callable process is invoked by a call activity step. The input specification for a callable process determines the data that flows into the start none event in the callable process when it is triggered by a call activity. The output specification for a callable process determines the data that flows out of the callable process and back to the call activity that triggered it. The entire pipeline is automatically sent back to the call activity.

Properties Page	Property	Description
		Adding, editing, and removing inputs and outputs for a global process is done the same way as it is for steps.
Server	**Integration Server Name**	The Integration Server used to get services and documents for the process.

Process Model Naming and Versioning

The name used to create a new process model is restricted to ASCII-numeric characters as this value is used in the **Process ID**.

The **Process Display Name** of a process model can be different from the name initially used to create the process model and supports multibyte characters. The process display name is used in the **Solutions** tab in Designer and the Business Console user interface. Modifying the process display name does not change the **Process ID** in any way. When searching for processes in the Business Console user interface, you can use both the process display name and the process ID.

The folder structure of process models is based around the process project, process ID, and the process model version. Each process model version can have a different process display name, but the filename in the local Designer workspace will remain the same for all versions. Each model version has a separate folder and can be checked in into source control separately with the process model files for all versions having the same filename.

Starting a New Process Instance

To create an action to start a new process instance

1. In the Rules Explorer view, right-click and then click **New > Action**in the context menu.
2. On the Process Action Type page, click the type of process action you want the rule to invoke.
3. Select **Start a new process instance**.
4. Click **Next**.
5. On the New Process Action page, select one or more process model names to start a new instance of those models.
6. In the **Integration Server Name** list, select the Integration Server where the process is defined. Click **Next**.

7. On the Document Type Selection page, select the IS document type to use as input to the process instance. Click **Next**.
8. On the Process Action Default Values page, specify any default data field values you want to include. These values are overridden when data is provided from an associated process. Click **Next**.
9. On the Process Action Return Value page, select a return value check box as required.
10. Click **Finish**.

Joining a Running Process Instance

To create an action to join a running process instance

1. In the Rules Explorer view, right-click and then click **New > Action**in the context menu.
2. On the Process Action Type page, click the type of process action you want the rule to invoke.
3. Select **Join a running process instance**.
4. Click **Next**.
5. On the New Process Action page, select one or more process model names to join a running instance of those models.
6. In the **Integration Server Name** list, select the Integration Server where the process is defined. Click **Next**.
7. On the Document Type Selection page, select the IS document type to use as input when joining the process instance. Click **Next**.
8. On the Process Action Default Values page, specify any default data field values you want to include. These values are overridden when data is provided from an associated process. Click **Next**.
9. On the Process Action Return Value page, select a return value check box as required.
10. Click **Finish**.

Disaster Recovery and Exception Handling

Disaster Recovery

Dynamic Business Orchestrator stores the information necessary to run a step in the cache until the step completes successfully. The StepInput cache is used for automatic disaster recovery. The StepInput cache exists for as long as the process step is being

executed and is not related to the execution of the whole process instance. A running step is kept in the cache in case of unforseen events, for example power failure, and the step data is used for disaster recovery after the system becomes available.

Exception Types

The exceptions that may occur during the process execution in Dynamic Business Orchestrator fall in one of the following categories:

- "Fatal Exceptions" on page 58
- "Unhandled Exceptions" on page 58
- "Handled Exceptions" on page 58

Fatal Exceptions

A fatal exception occurs when the process model has a design flaw and cannot be resolved regardless of how many times the instance is resubmitted, for example, the model violates a BPMN execution rule.

In the event of a fatal exception, the status of the step that causes the exception is changed to **Failed**, the process instance status is changed to **Failed**, and no further action can be taken. The process instance is cleared from the Dynamic Business Orchestrator cache leaving only an Audit Trail of the instance for future analysis.

Unhandled Exceptions

An unhandled exception occurs in a step that has no boundary error event handler associated with it. The BPMN implementation is created in a way that process instance execution must be completed even with errors in the instance. The process designer has to consider the various error conditions and design the process model accordingly.

However, most error conditions cannot be anticipated and when a process step fails, that step may be required to run again. When this happens, the step must be restarted. To allow step restart, the process instance must be blocked in case of an unhandled error.

When an exception occurs and there are no error handlers in the process model, the exception is an unhandled exception. In this case, the step status is changed to **Failed** and the process instance status is changed to **Blocked**. You restart a blocked instance using Dynamic Business Orchestrator services from the webMethods Business Console user interface. A blocked process instance is still considered **Running**.

Handled Exceptions

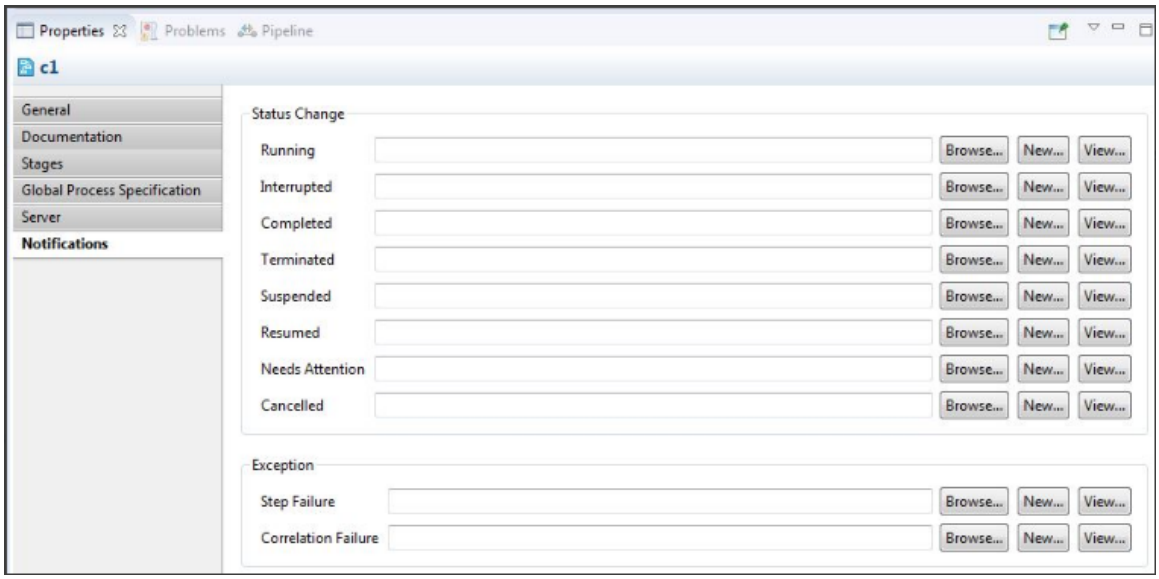
An exception is considered a handled exception when there is a boundary error event in the process model to process the exception. Handled exceptions occur when the model was drawn to anticipate failure and the execution pattern changes as a result of the error handler. Depending on the error handler, the process may continue its execution.

Handled exceptions do not interrupt or stop the execution of the process. The status of the step that causes the exception is changed to **Failed** (or **Interrupted**) and the process instance status continues to be **Running**. A step resulting in a handled exception cannot be restarted by definition.

Triggering Custom Error Handling Without Altering the Control Flow

You can trigger custom error handling that does not alter the control flow in a process. A *notification* is used to select a specific service to invoke when one of several status changes or exceptions occur. The specified service runs without changing the process or step status along the way.

For example, if you want a service to run when the process status changes to Completed, you associate a service with the Completed status. The custom error handling can be set up on the Notifications tab in the process's properties in Designer.



Process Validation

Dynamic Business Orchestrator process models are validated at design time and process instances are validated at runtime. Processes can also be validated in the Dynamic Business Orchestrator API.

VI Administering Process Models

Working with Stages and Milestones

You can create, delete, and modify process stages in Monitor.

You must have a BPM or a BPM and BAM server environment selected in the Server list at the top of the Process Instances page before you can add stages in Monitor.

You can display a stage's start milestone and stage end milestone in the Process Diagram window. Only one stage can be displayed in the Process Diagram at any time.

Adding a Stage

Note: If you leave the **Stages** tab while adding a stage and before you have clicked **Save**, your changes will be discarded.

To add a stage

1. On the Business Processes page, locate and edit the process model that you want to work with.
2. In the Process Stages and EDA Events window, click the **Stages** tab.
3. Click **Add Stage**. A new row appears in the stage list, populated with default information.
4. Configure the following fields to define the stage.

Note: Any data entry validation errors are displayed within the stage row.

Column	Description
Name	Type a name for the stage. The **Name** is not editable after you click **Save**. If you want to rename a stage, you must delete it and then recreate it with the new name. There is an 80-character limit for the stage name when double-byte characters are used in an IBM DB2 database. If you are not using DB2, or if your characters are single byte, then the stage name is limited to 255 characters.
Description	Optional. Type a description of the stage.

Column	Description
Start Milestone	Click the list and select a milestone. Optionally, you can type characters in the text box to filter the list. The **Start Milestone** and **End Milestone** selections must be different. Click the list to the right of the milestone selection, and click **Start** or **Complete** to specify the start or the completion of the selected milestone.
End Milestone	Click the list and select a milestone. Optionally, you can type characters in the text box to filter the list. The **Start Milestone** and **End Milestone** selections must be different. Click the list to the right of the milestone selection, and click **Start** or **Complete** to specify the start or the completion of the selected milestone.

Condition	■ Select **<** (less than) or **>** (greater than). Default is **<**.
	■ Enter a positive whole number. The maximum supported values are as follows:
	  ■ 2,777,777 hours
	  ■ 166,666,666 minutes
	  ■ 9,999,999,999 seconds
	  ■ 9,999,999,999,999 milliseconds
	■ Default is **1**.
	■ Select **weeks**, **days**, **hours**, **minutes**, **seconds**, or **milliseconds**. Default is **hours**.
	
	The result is a condition. If the condition specifies <, then the stage is breached when the cycle time exceeds the specified time period. If the condition specifies >, then the stage is breached when the cycle time is less than the specified time period. For example:
	
	`< 1 hours` means that the stage must complete in less than 1 hour or a `ProcessStageBreached` event will be emitted.

Stop Tracking On Breach	Stops stage processing for all remaining stages in the process instance when a stage breach occurs in this stage, and only one stage breached EDA event is emitted. Remaining stages are not tracked and will be shown as Incomplete in Monitor. The check box is cleared by default.

5. Click **Save**.

Modifying a Stage

Note: If you leave the **Stages** tab while modifying a stage and before you have clicked **Save**, your changes will be discarded.

You cannot modify a stage name. If you want to rename a stage, you must delete it and then recreate it with the new name.

All other stage and milestone information can be modified as described in "Adding a Stage" on page 62.

Deleting a Stage

Note: If you leave the **Stages** tab after deleting a stage and before you have clicked **Save**, the deletion will be discarded.

To delete a stage

1. On the Business Processes page, locate and edit the process model that you want to work with.
2. In the Process Stages and EDA Events window, click the **Stages** tab.
3. Click the option button next to stage name for the stage you want to delete. To clear your selection, click the option button again.
4. Click **Delete**.
5. Click **Save**.

Synchronizing Stage Settings

You can create, modify, and delete stage settings in two locations:

- In Software AG Designer, on the **Stages**page in the Properties view.
- In webMethods Monitor,on the Business Process administration pages in My webMethods.

Changes to the stage settings from webMethods Monitor's Business Process administration pages are saved to the Process Audit database. The saved changes overwrite the existing setting details in the database. As a best practice, you should ensure that your stage settings are always synchronized between the two locations.

Important: Deleting a step that is contained in a process stage without first synchronizing the stage settings with the database can lead to the process being out of sync with edits done in Monitor. Always click the stage settings

Synchronize button immediately before you delete any steps in Designer that are contained in a stage.

To synchronize stage settings:

1. Open the process model and in the process editor, click the design canvas to select the entire process.
2. In the Properties view, click the **Stages** tab.
3. Click **Synchronize**. Designer retrieves the stage settings from the Process Audit database and applies them to the process model.
4. Click **Save** to save the stage settings to the local workspace.

To make your changes available to the runtime, you must upload the process model. Uploading the process overwrites the existing stage settings in the Process Audit database.

Deleting Unused Process Models

If a process model has not been used, you can delete information about the process model from the Process Audit Log database and the Monitor display. Before you can delete a webMethods-executed process model version, you must first disable that process model. You can delete any type of process (webMethods-executed, externally executed, or integration process) as long as that process has never been used for a process instance.

Note: To delete a process model you must have My webMethods Server administrator privileges.

To delete unused process models

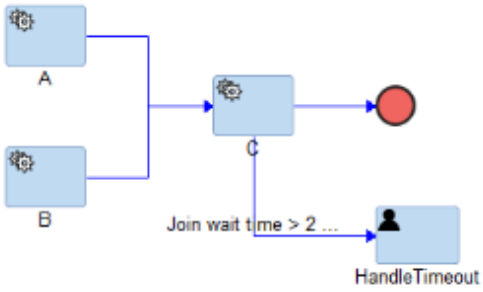
1. On the Business Processes page, search for the process model you want to delete.
2. In the search results, select a check box for each process model you want to delete.
3. Click **Delete**.

VII Migrating Classic BPM Process Models to Dynamic Business Orchestrator

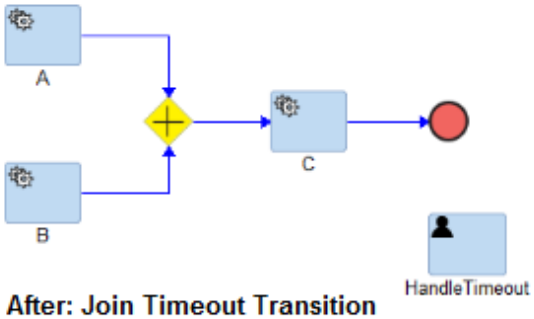
BPM Transitions Not Supported by Dynamic Business Orchestrator

Join Timeout

In classic BPM, you can configure a timer on a join expression, which instructs the runtime to take the join timeout transition when the configured amount of time has elapsed. In Dynamic Business Orchestrator, a gateway simply waits for all inbound sequence flows to arrive, therefore the join timeout concept does not exist in Dynamic Business Orchestrator. The migration utility eliminates the join timeout transition from migrated models, which will "orphan" the steps that were connected to that transition in the classic model. This requires a manual change to the migrated model to achieve the desired execution behavior.

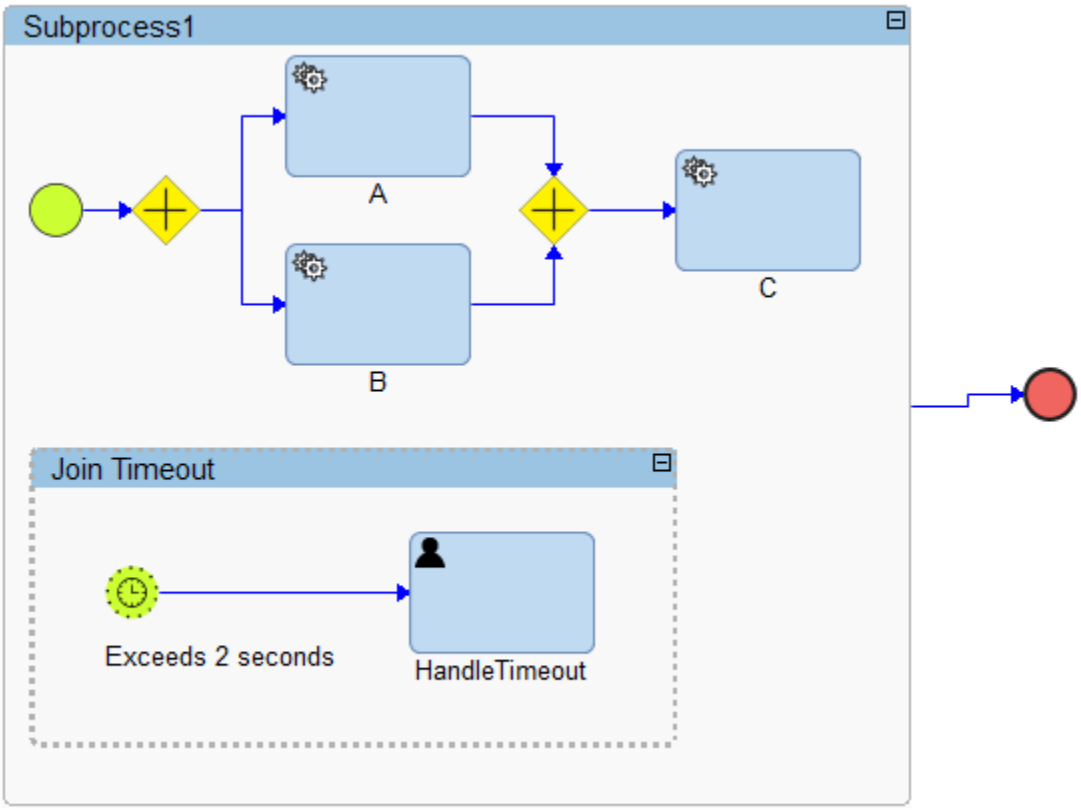


Before: Join Timeout Transition



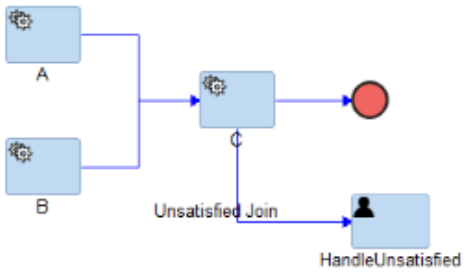
After: Join Timeout Transition removed

To resolve the orphaned steps that were once connected to the join timeout transition, you can simply remove the steps. Another option is to use a Timer Event Subprocess to achieve an equivalent pattern. The following image is an example of how a Timer Event Subprocess is used to implement the classic join timeout pattern.

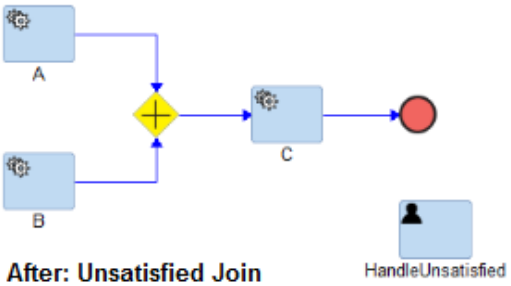


Unsatisfied Join

In classic BPM you can use Unsatisfied Join, which is a special transition that executes when a Join Expression fails to be satisfied. Dynamic Business Orchestrator does not provide this special transition, because a properly designed model should never result in a gateway that fails to execute. The migration utility eliminates the UnsatisfiedTimeout transition from migrated models, which will "orphan" the steps that were connected to that transition in the classic model. This requires a manual change to the migrated model to achieve the desired execution behavior.



Before: Unsatisfied Join Transition



After: Unsatisfied Join Transition removed

A properly designed model should never look for a failing join to execute correctly, so the recommended approach is to examine the gateways used in the model to ensure

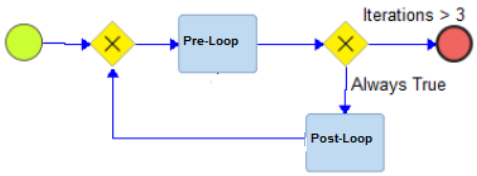
they correctly describe the diverging and converging behavior. Once you make sure the gateways in the model are set up correctly, remove the orphaned steps that were connected to the unsatisfied join transition.

Step Iterations Exceeded

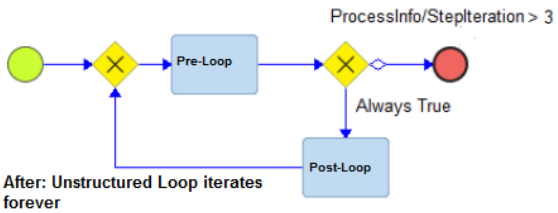
In classic BPM you can use Step Iterations Exceeded, which is a special transition that executes when a particular step had been executed a configured number of times. The most common use of this transition isto exit a what is known as an unstructured loop. In classic runtime, if the step exceeded the configured number of iterations, the Step Iterations Exceeded transition would execute and prevent the execution of any other transitions for that step. This transition type is not directly supported by Dynamic Business Orchestrator, but the migration utility replaces it with a conditional transition that is true when the step exceeds the configured number of iterations.

Important: Dynamic Business Orchestrator does not give special treatment to any particular sequence flow and evaluates all conditional sequence flows for a particular step. When a migrated classic model relies on the Step Iterations Exceeded transition to ignore other conditional transitions, that model needs to be manually modified to achieve the desired result.

The following example shows a classic model with an unstructured loop that is configured to exit after 3 iterations and the same model after migration. The Step Iterations Exceeded transition is replaced by a normal, conditional sequence flow that examines the StepIteration. This transition executes as expected, but in the classic model the transition labeled "Always True" will be prevented from executing, while Dynamic Business Orchestrator executes that transition. This unstructured loop would executed infinitely following migration. This is one example of the importance of inspecting the migrated model when using the Step Iterations Exceeded transition.



Before: Unstructured Loop Exits after 3 iterations



After: Unstructured Loop iterates forever

The use of unstructured loops is not advised. If they are absolutely necessary, it is important to inspect the conditional sequence flow that governs the loop and modify the conditions in the migrated model to ensure that the loop will properly exit and not just rely entirely on the Step Iterations Exceeded transition. Alternatively, you can replace the unstructured loop with a subprocess and structured looping.

The Model Package

Classic BPM creates a package that contains various generated assets associated with a model. For example, the subscription triggers, transition triggers, and wrapper services are generated into various folders in this model package. A classic BPM model is

dependent on this package for these various assets and needs to be deployed along with the classic model.

The Dynamic Business Orchestrator does not have a requirement for a model package, which is one less dependency during deployment. Classic BPM model generates wrapper services in the model package, while a Dynamic Business Orchestrator stores mapping internally in the model. Classic BPM model generates triggers in the model package, while Dynamic Business Orchestrator automatically creates all necessary triggers for all model-versions in a single package known as WmDBOAssets.

Migration of Model Properties

Unless specifically mentioned, Model Properties are directly migrated to the DBO model without modification.

KPIs

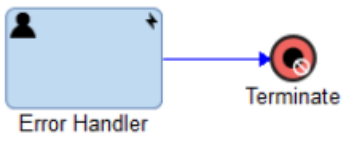
The configuration of KPIs are not supported in Dynamic Business Orchestrator.

Error

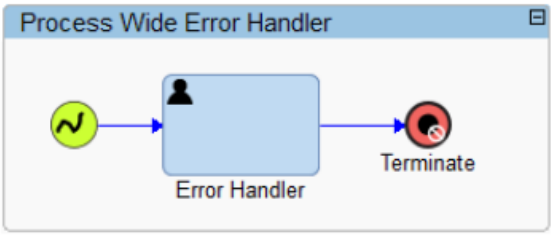
The Error property in classic BPM designates a step (or a sequence of steps) to execute when an unhandled exception occurs in a model. This property does not exist in Dynamic Business Orchestrator and is migrated to an Error Event Subprocess.

Important: The Error Event Subprocess is interrupting by definition and this may not be the desired behavior for classic models that employed an error handler that was not interrupting.

Manual modifications to the model might required to get the desired behavior at run-time.



Before: Error handling step



After: Error handling encapsulated in Error Event Subprocess

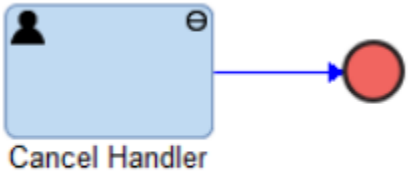
If the designated Error Step has inbound sequence flow, it cannot be migrated to an Error Event Subprocess as this breaks the definition of an Event Subprocess. In such cases the steps are migrated normally, but the "Error" designation is removed and the model requires manual modification to achieve the desired run-time behavior.

Before: Error Handler with Inbound Sequence Flow

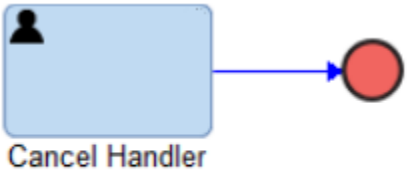
After: Error designation removed, manual modification required

Cancel

The Cancel property in classic BPM designates a step (or a sequence of steps) to execute when an instance is cancelled. This is not supported in Dynamic Business Orchestrator. If a migrated model has a Cancel step configured, the step is migrated normally but it does not have the "Cancel" designation and is not executed when the instance is cancelled.



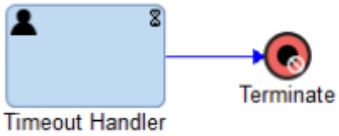
Before: Cancel Handler



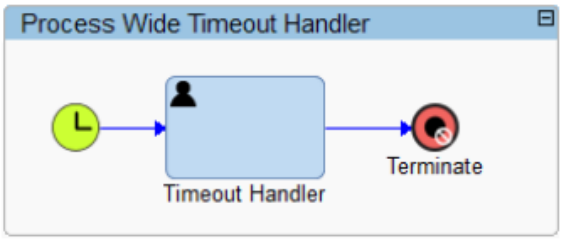
After: Cancel desingation removed

Timeout

The Timeout property in classic BPM designates a step to execute when a configured timeout occurs. There is no concept of a Process-Wide timeout Step in Dynamic Business Orchestrator, but a Timer Event Subprocess closely resembles this pattern. The Timeout steps are migrated to a Timer Event Subprocess with non-interrupting behavior. The migrated model must be inspected and manually modified if the desired behavior of the Timeout step is interrupting.

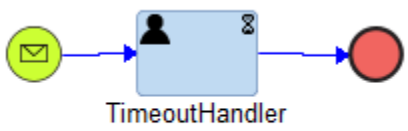


Before: Timeout handler



After: Timeout handler encapsulated in Timer Event Subprocess

If the designated Timeout Step has inbound sequence flow, it cannot be migrated to a Timer Event Subprocess as this breaks the definition of an Event Subprocess. In such cases, the steps are migrated normally but the "Timeout" designation is removed and the model requires manual modification to achieve the desired run-time behavior.



Before: Timeout Handler

After: Timeout designation removed

Optimize Locally

This checkbox in classic BPM instructs the runtime to transition from step to step without publishing transition documents. In Dynamic Business Orchestrator, step transitions are not achieved through the publishing of documents and this option is not migrated.

Express Pipeline

This classic BPM option results in a named set of pipeline elements to be written into the process fragment for a particular step. When this set exists, the classic runtime preserves these named elements in the pipeline. In Dynamic Business Orchestrator, the pipeline is not manipulated in this way. Mapping and service tasks are the only way to modify the pipeline. The migrated model might require manual modification to achieve the desired pipeline.

Volatile Transition Documents

This classic BPM option instructs the runtime to use volatile or persistent transition documents. There are no transition documents in Dynamic Business Orchestrator and this option is not migrated.

Volatile Tracking

This classic BPM option instructs the runtime to store process execution information in memory (volatile) or the Process Engine database (persistent). In Dynamic Business Orchestrator, all process instance execution information is stored in the Dynamic Business Orchestrator database and this property is not migrated.

Minimum Logging Level

This classic BPM option customizes how much Audit Logging is performed at run-time. Dynamic Business Orchestrator does not allow the configuration a logging level and this property is not migrated.

Emit Process-specific Predefined EDA Events

This classic BPM option customizes which EDA events are emitted during process execution. The Dynamic Business Orchestrator does not emit events for transitions, but emits a standard set of Events during execution and there is no configuration required.

Steps Enabled For Resubmission

This classic BPM option instructs the runtime to log Audit Data during step execution to facilitate the resubmission of that step when an error ocurrs. Dynamic Business Orchestrator automatically provides this capability and this option is not migrated.

RosettaNet

This option is not migrated.

Enable deprecated error handling behavior

This option is not migrated.

Default Deployment Values

There are two checkboxes in classic BPM that instruct Deployer to enable the deployed model for execution and/or analysis. These checkboxes do not exist in Dynamic Business Orchestrator and are not migrated. Deployer can be instructed to enable deployed models.

Distributed Processing

Classic BPM supports a distributed architecture where specific steps in a model execute on a particular Integration Server. This is done by assigning logical servers to steps in the model. The model is generated into a fragment for each logical server. Each fragment only references steps associated with a particular logical server. At run-time, only the steps known to a logical server will be executed for a process instance, effectively distributing the execution of the model across any number of Integration Servers.

The use of this feature is common, but requires a fairly complex messaging configuration to ensure that transition events are handled by the appropriate BPM node. This configuration is can cause issues and difficulties during installation and when configuring new BPM instances in an environment.

Dynamic Business Orchestrator treats the model definition as a single unit that cannot be fragmented. The concept of logical servers in a Dynamic Business Orchestrator model does not exist. Every Dynamic Business Orchestrator node has full visibility to the model and requires that the model can be executed on a single Dynamic Business Orchestrator node. In other words, all Integration Server namespace artifacts referenced by a Dynamic Business Orchestrator Process model have to exist on the Integration Server executing the model.

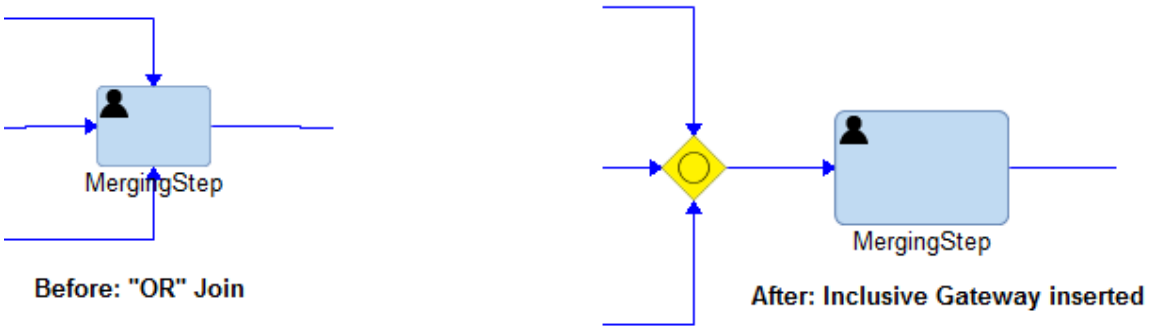
Prior to executing a classic model that uses the distributed architecture migration utility, the machine hosting Designer must be connected to every Integration Server referenced by the logical servers in the model. This is necessary so the migration utility can query those assets for the creation of maps that are stored in the model.

Upon migration, the Dynamic Business Orchestrator model will no longer have references to logical servers and will expect that every model dependency exists on every Integration Server node that is designated to execute the model.

Insertion of Gateways for Multiple Inbound Sequence Flow

In a classic BPM model, you can have multiple inbound sequence flow into steps that are not gateways. The "converging behavior" of these sequence flows is defined by the join type (Or, And, Complex, Unsynchronized Or) configured in the Join properties panel for that step. In Dynamic Business Orchestrator, only gateways can have multiple inbound sequence flow, therefore it is necessary for the migration utility to insert a gateway to replace the multiple inbound sequence flow. Insertion of Gateways will be one of the most obvious changes to your model as the migration utility inserts gateways in the diagram to preserve the overall look. The resulting gateway will dramatically improve the readability of the model notation by better illustrating what the converging behavior of the sequence flow should be.

For example, if your step was governed by an "OR" Join, the migration utility will insert an Inclusive gateway and reassign the sequence flow from the source steps to the gateway and then from the gateway to the target step. The following image is an example of how this looks in a classic and the migrated Dynamic Business Orchestrator model.



Depending on the type of join, the following gateways are inserted for multiple inbound sequence flow:

- An Inclusive gateway is inserted for an "OR" join.
- A Parallel gateway is inserted for an "AND" join.
- A Complex gateway is inserted for a "Complex" join.
- An Exclusive gateway is inserted for an "Unsynchronized Or" join.

Insertion of Gateways for Multiple Outbound Sequence Flow

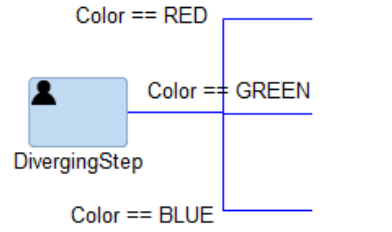
In a classic BPM model, you can have multiple outbound sequence flow from steps that are not gateways. In Dynamic Business Orchestrator, only gateways can have multiple

outbound sequence flows, therefore it is necessary for the migration utility to insert a gateway to respect the "diverging behavior" of the step. Insertion of gateways is one of the most obvious changes to your model as the migration utility inserts gateways in the diagram to preserve the overall look. The resulting gateway improves the readability of the model notation by illustrating what the converging behavior of sequence flow should be.

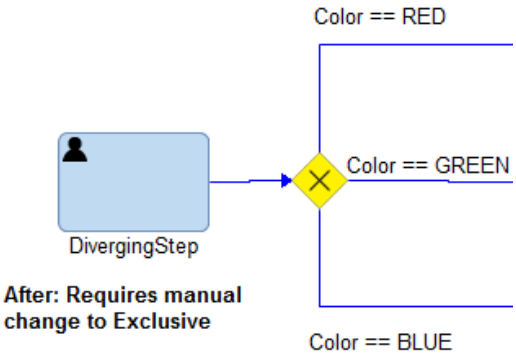
The inserted gateway depends on the structure of the outbound sequence flow. For example, if the sequence flow had no transition conditions configured, the migration utility will insert a Parallel Gateway, reassign sequence flow from the source step to the inserted gateway and then reassign sequence flow from the gateway to all target steps. The following image is an example of how this looks in a classic and the migrated Dynamic Business Orchestrator model.

When there are one or more transition conditions present, an Inclusive Gateway is inserted by the migration utility. This is the behavior of the migration utility, because it does not know if the diverging behavior was intended to be exclusive, inclusive, or complex. In this case, you are advised to manually review your model and inspect the transition conditions for the inserted gateway. If the transition conditions are characterized by "at least one path will be true", that is inclusive diverging behavior and the inserted inclusive gateway would not require modification.

When the transition conditions are characterized by "exactly one will be true", that is exclusive diverging behavior and the inserted inclusive gateway should be manually changed to exclusive.

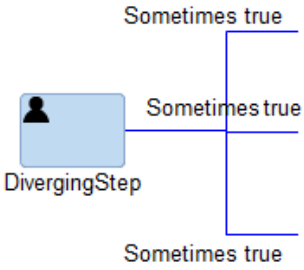


Before: Exclusive Diverging Behavior

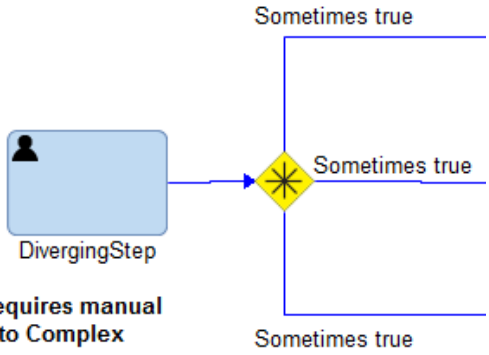


After: Requires manual change to Exclusive

When neither of the previous examples describes the intent of the conditional sequence flow (not mutually exclusive and no guarantee that at least one condition will be true), this fits the description of complex diverging behavior and the inserted inclusive gateway should be manually changed to a complex gateway.



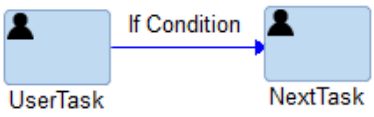
Before: Complex Diverging Behavior



After: Requires manual change to Complex

Insertion of Gateways for Single Conditional Outbound Sequence Flow

In a classic BPM model, you can configure a transition condition on a single outbound sequence flow from a step. Dynamic Business Orchestrator does not permit transition conditions on any step type except gateways, so it is necessary to insert a complex gateway between the source step and the target step and transfer the condition to the outbound sequence flow on the inserted gateway.



Before: Transition Condition on Step Sequence Flow

After: Gateway inserted, transition condition transferred

Generated Flow Services

In classic BPM, most step types result in the generation of flow services as a convenience for performing data mapping prior to (or after) step execution. These flow services are known as "wrapper services" to distinguish them from ordinary Flow Services. Designer generates these wrapper services into the namespace of the model package and the user is welcome to modify these wrapper services as they see fit. The wrapper services are considered a dependency of the model and must be deployed as part of BPM model deployment.

In DBO, Designer does not generate wrapper services. Instead, one may configure data mapping directly into the model using a built-in mapping Flow Editor. This eliminates the need for extneral dependencies on flow services to perform mapping. It is still permissible to create external flow services and invoke them as part of a Service Activity, but it is recommended to configure all mapping within the model.

During migration, the utility will make every effort to move the contents of wrapper services to the internal mapping structures associated with a step. The following table provides the specifics for each step type.

Migration of Step Types

Unless specifically mentioned, classic BPM step types are migrated to the same step type in Dynamic Business Orchestrator and there is no difference in runtime behavior.

Service Tasks

The classic BPM Service Task is implemented with a wrapper service that is generated by Designer and located in the model package. The intent of the wrapper service is to provide a place to do mapping and/or build a complex service implementation. Depending on how the wrapper services are created, the wrapper service could contain input mapping, execution of logic, and output mapping all in one place. It is possible to invoke a custom service from the generated wrapper service and the invoked custom service lives outside the model package.

A Dynamic Business Orchestrator Service Task is implemented by 3 components:

1. Input Mapping
2. The Service
3. Output Mapping

If a wrapper service contains more than one invoke step, Dynamic Business Orchestrator configures the migrated service task to use the wrapper service (just as classic BPM does). If the wrapper service contains a single invoke of another service, the migration utility configures the migrated service task to use the invoked service, eliminating the

need for the wrapper service in the deployed solution. In either case, you can embed your Input and Output mapping directly into the model.

Send Tasks

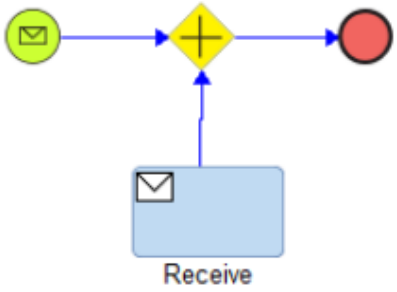
Send Tasks are not supported by Dynamic Business Orchestrator. They are functionally replaced by Throwing Message Events.

A classic BPM Send Task with outbound sequence flow is migrated to a Throwing Message Event. When a Send Task has no outbound sequence flow, it is migrated to an End Message Event

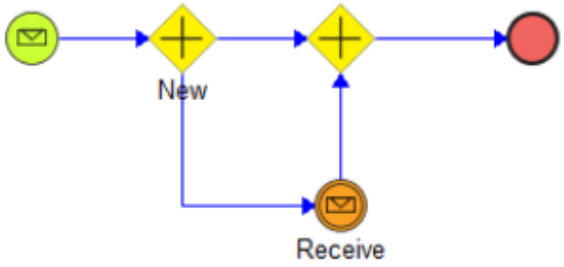
Receive Tasks

Receive Tasks are not supported by Dynamic Business Orchestrator. They are functionally replaced by Catching Message Events.

If a Receive Task has no inbound sequence flow, it is migrated it to an Intermediate Catching Message Event along with an inserted sequence, connecting the Start Event in scope to the migrated Message Event. A Gateway is also inserted to deal with the multiple outbound sequence flow from the Start Event. The image below illustrates this behavior for a Receive Task in a top-level process.

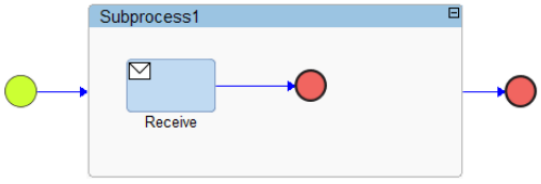


Before: Receive Task w/no Inbound Sequence Flow

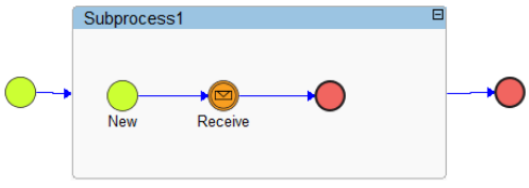


After: Migrated to Intermediate Message Event and Sequence flow from Start Event

If a Receive Task without sequence flow appears in a subprocess, a Start None Event is inserted to connect to the Intermediate Message Event.



Before: Receive Task w/no Sequence Flow in a Subprocess



After: Start None Event inserted and Sequence Flow connecting to the Intermediate Message Event

If there is inbound sequence flow and outbound sequence flow, the step is migrated directly to an Intermediate Message Event.

Deprecated webMethods Subprocess

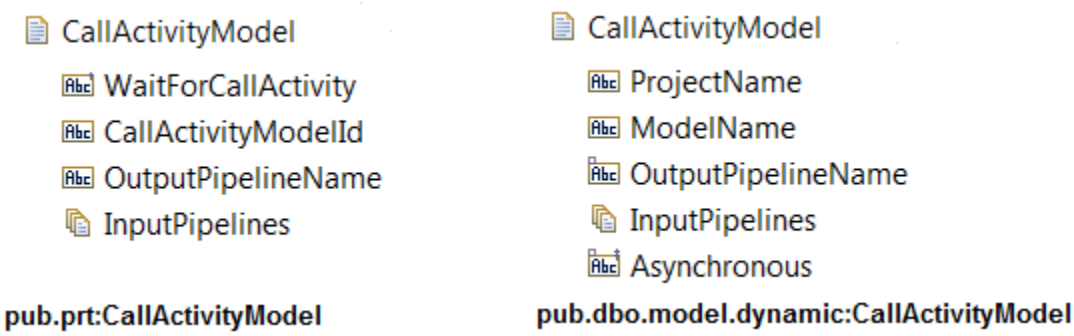
Classic BPM supports two types of Subprocess: webMethods Subprocess and BPMN Subprocess. The webMethods Subprocess was deprecated in BPM Version 8.2 and is not supported by Dynamic Business Orchestrator. A webMethods Subprocess is migrated to a BPMN Subprocess, but it is possible that the migrated subprocess will execute differently in Dynamic Business Orchestrator than it did in classic BPM. You must manually verify that your migrated subprocess behaves as desired and make changes as necessary.

Deprecated webMethods Referenced Subprocess

webMethods Referenced Subprocess was deprecated in BPM Version 8.2 and is not supported by Dynamic Business Orchestrator. A webMethods Referenced Subprocess is migrated to a call activity, but the migrated result might not execute as it did in classic BPM. You must manually verify that your migrated webMethods Referenced Subproces behaves as desired and make changes as necessary.

Dynamic Call Activity

Compared to classic BPM, the document required to setup the execution of a Dynamic Call Activity is modified in Dynamic Business Orchestrator. You must manually modify the Input and Output Mapping for a migrated Dynamic Call Activity.



End Terminate

The End Terminate step in classic BPM is configured to end the process instance with a particular status (Copmpleted, Cancelled, Failed). In Dynamic Business Orchestrator, an End Terminate has a single definition: to abnormally terminate the process instance. A classic End Terminate Event is migrated differently depending on the configured status as follows:

- A configured Completed status results in an End None Event after migration.

- A configured Cancelled status results in an End Terminate Event after migration.
- A configured Failed status results in an End Terminate Event after migration.

Note: Dynamic Business Orchestrator does not support model notation to cancel a process instance. A classic End Terminate Step, configured for "Cancelled", is migrated to an End Terminate Event, which has a different runtime behavior. In Dynamic Business Orchestrator, a process instance can be cancelled using the service pub.dbo.instance.control:cancel.

Migration of Step Properties

Implementation, Integration Server Name

This value is ignored during migration, because Dynamic Business Orchestrator does not support distributed execution of the model.

Implementation, Allow Parallel Execution

The state of this checkbox is ignored during migration, because Dynamic Business Orchestrator has built-in protection against parallel step execution.

Implementation, Service, Web-Service

If this radio button is selected in a classic BPM Service task, the value is ignored during migration, because Dynamic Business Orchestrator only supports Integration Service services.

Implementation, Generated Service Name

This value is ignored during migration, because Dynamic Business Orchestrator does not generate wrapper services for a service activity task.

Implementation, Retry Count & Retry Interval

These values are ignored during migration, because Dynamic Business Orchestrator does not support the automatic retry of service tasks.

Implementation, Send / Receive Protocol

Dynamic Business Orchestrator currently supports Universal Messaging protocol for message receipt and publishing. Classic BPM supports Native Pub/Sub, JMS, and Simple-Service (request-reply) protocols. The protocol will be migrated to Universal Messaging for each Catching and Throwing message event in the classic model.

Joins Properties

The Join properties for a classic BPM step are not migrated to the Dynamic Business Orchestrator model, because a gateway is inserted during migration when a step contains the join. These properties no longer appear in Dynamic Business Orchestrator.

This includes the Join Type, Join Timeout, and the deprecated Suppress Join Failure properties.

Ignore dead path notifications (Deprecated)

This deprecated property is not migrated to the Dynamic Business Orchestrator model. Dead path notifications are managed exclusively by Dynamic Business Orchestrator.

Using the Migration Report View

Once you migrate a 9.x process model to a 10.x Dynamic Business Orchestrator process, the Migration Report View gives you information on the migration. The top section of the migration report view provides the following information:

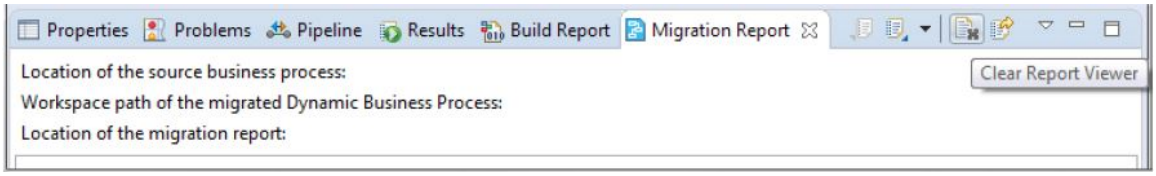
- Location of the source business process
- Workspace path of the migrated Dynamic Business Process
- Location of the migration report

Each field is a clickable link that takes you to the corresponding item.

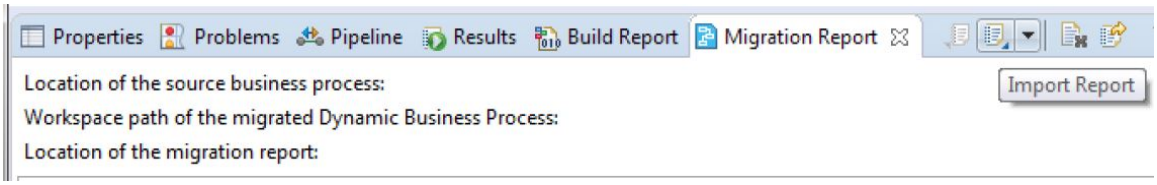
The migration file name includes the name of the process being migrated as well as a date-timestamp of the migration. For example, the file name loopContainer_MigrationReport2018_08_03 021140PM.log indicates that the process "loopContainer" was migrated on August 3, 2018 at 2:11:40PM.

Note: Editing the migration report file updates the operating system timestamp for the file. This allows for a better tracking of the migration timeline.

You can clear the migration report by clicking the **Clear Report Viewer** button:



Migration projects are usually ongoing. In such cases you can open a previous migration report by clicking the **Import Report** button:



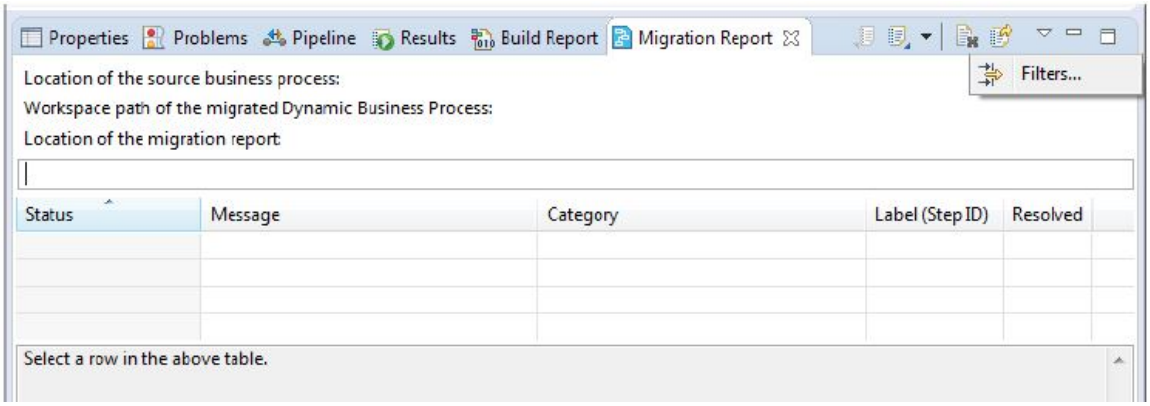
Migration Report Contents

Migration is a complex process that requires you to perform post-migration steps. Many classic BPM structures are migrated directly to Dynamic Business Orchestrator, but

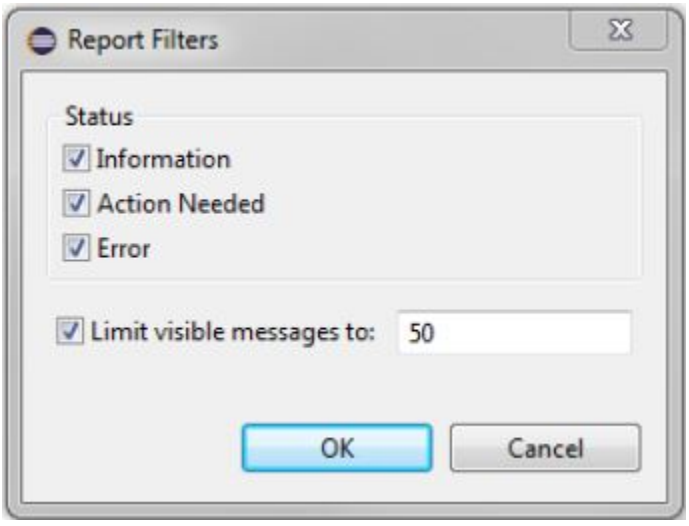
migration challenges occur, depending on the migrated process. The migration often results in the altered look of the model and you might be required to modify the model after the migration. In some cases, the process structure might not be migrated at all.

The migration report view contains a table of migration details, where each row describes both the results of the migrated item and any post-migration actions that are needed.

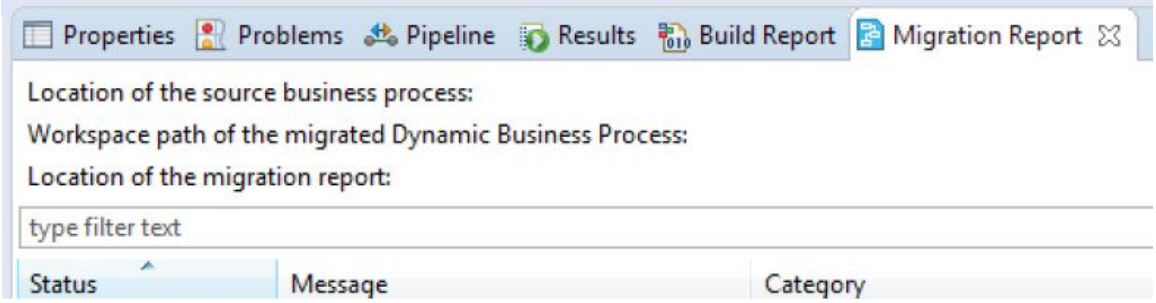
You can filter the rows in the migration report table by customizing the filters that are shown:



The **Report Filters** dialog allows you to specify the items that you want to see in the table, with all status types being displayed and the visible messages limit being set to 50 by default.



You can also type filter text above the migration report table, which applies text filter to all columns except Resolved.



Note: Each column can be sorted in either direction. Double clicking on a row will highlight the affected step in the canvas. If the modern process is not already opened in the editor, then the double click will first open the process.

The migration report table has five columns, each displaying different information:

The Status Column

The status column contains one of three statuses and an icon associated for that status:

- *Information* The green ball icon, , is used for information items. This status is used to highlight a change that was made during migration to bring the migrated process model in line with the requirements of a Dynamic Business Orchestrator process model. For example, BPMN standards only allow multiple inbound sequence flows into a gateway step and an information message is provided to describe that change. There is no action needed for this item type.
- *Action needed* The yellow triangle icon, , is used for items that you need to address. Items that are marked as "Action needed" require action on your part. Most migration items fall into this, or the Information category. For example, Referenced Subprocesses are deprecated and are replaced during migration. When migrating a process that contains such subprocess, you will see a message similar to “Referenced Subprocess is a deprecated structure. It was migrated to a BPMN Call Activity, but you will need to manually modify the child process model in order for it to behave correctly as a callable child.”
- *Error* The red stop sign icon, , is used for items that caused an error during migration and require your attention. Items that are marked as "Error" require action on your part. For example, if the generated wrapper services in the model package are not available during migration, then any maps associated with that migration will be lost. For that situation you will see a message similar to “The flow.xml file for step "S8" was not found on the logical server "Default" under "PEBVT_Base \ns\PEBVT_Base.ANDjoin.ANDjoin_1.Default:S8\flow.xml". In such cases any mappings done in your source process cannot be migrated to your target process. Some possible reasons are:
  - You did not build and upload the source process
  - You changed the Generated Package Name value in the source process's properties, but did not subsequently build and upload the process
  - The flow.xml file is located on a networked IS that cannot be reached at this time.

The Message Column

The message column contains a brief description of the issue reported. And that description should be used along with the migration documentation to resolve the issue.

The Category Column

The category column contains a unique identifier, enabling you to group issues together and make it easier to navigate.

The Label (Step ID) Column

The label (Step ID) column contains the step's label followed by the step's ID in parenthesis. That is intended to aid you in locating the step in your process. You can also double-click on the row to highlight the step in your process on the canvas.

The Resolved Column

The resolved column contains an editable check box. During your migration project, as you resolve items, this column allows you to indicate that.

VIII Monitoring and Controlling Process Instances

Controlling Process Instances

In the webMethods Business Console web user interface you can apply certain actions to a running, suspended or needs-attention process instance to control the process flow.

Depending on the current status of the process instance, different actions are available for that instance.

Note: Business Console displays only valid actions for the process instance. When the current status of the process instance does not permit any actions, no valid actions are displayed in the drop-down menu.

Cancelling a Process Instance

You can cancel a running or suspended process instance, which prohibits further execution. Cancelling a process instance results in all running steps being cancelled and the process instance status changes to Canceled.

To cancel a running process Instance:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Cancel**.
4. Click **Yes**.

Suspending a Running Process Instance

You can suspend a running process instance and later resume the execution of that process instance. Suspending a process instance changes the instance status to Suspended.

To suspend a running process Instance:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Suspend**.
4. Click **Yes**.

Resuming a Suspended Process Instance

You can resume a previously suspended process instance.

To resume a suspended process Instance:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Resume**.
4. Click **Yes**.

Restarting a Process Instance

You can restart a process instance when the instance is with Needs-Attention status. Restarting a process instance will restart all failed steps inside the process instance.

To restart a process Instance:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Restart**.
4. Click **Yes**.

Force Completing a Process Instance

You can force a process instance to Completed status, which cancels all running steps in the instance and the status of the process instance changes to Completed.

To force complete a process Instance:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Force Complete**.
4. Click **Yes**.

Changing the Model Version of a Running Process Instance

You can change the model version of process instances that are currently with status of Running, Suspended or Needs-Attention.

Important: When changing the model version of a process instance, only the current process instance is affected by the change. Any future instances of the process model will continue to execute on the original model version.

To change the model version of a running process instance:

1. In Business Console, on the **Processes** tab, select a process instance.

2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Change model version**.
4. Select a new model version.

   Note: All process models that are known by Dynamic Business Orchestrator are displayed. This includes model versions that are not currently activated.

5. Click **Change version**.

For more information on model version upgrade, see "Process Model Inflight Version Upgrade" on page 96

Process Instance Statuses

The following table shows the complete set of statuses for a process instance and the supported actions that you can perform to change the status of the process instance.

Status	Description	Available Actions
Running	The process instance is running.	Cancel, Suspend, Force Complete
Completed	The process instance is completed. This occurs when the track count reaches zero (normal) or if the instance was forced into completion. No further execution is permitted.	None.
Needs-Attention	One or more steps in the instance are failed or paused. Manual intervention is required for the instance to be running again. The instance is still considered active as other tracks in the instance may be executing normally.	Cancel, Resume, Force Complete
Cancelled	The process instance is cancelled. No further execution is permitted.	None.
Terminated	The process instance was terminated abnormally. The terminated status is a result of a fatal runtime exception or an end terminate event. **Note:**A fatal runtime exception indicates problems with the process design.	None.

Status	Description	Available Actions
Suspended	The process instance is suspended by a control action.	Cancel, Resume, Force Complete
Interrupted	The process instance was interrupted due to an interrupting boundary event or an interrupting event subprocess. No further execution is permitted.	None.

Controlling Process Steps

In the webMethods Business Console web user interface you can apply certain actions to a process step that is in a non-terminal state. Non-terminal state step status includes Running, Failed, Paused, Waiting and Redirected. Depending on the current status of the process step, different actions are available for that step.

You can apply actions to a process step from the following sources:

- The **Actions** menu of a process instance.
- The Process Diagram, by clicking the desired step.
- The Step Summary, by clicking on the down arrow on the **Actions** column.

You can edit or view the pipeline data of a step. Editing the pipeline data of a step is prompted when executing a step action that permits pipeline data editing. You can view the pipeline data of a step in read-only format from the Step Summary, by clicking **View Pipeline** for the desired step. Viewing of the pipeline is available for processes that are in a non-terminal state and for steps in that process that are in a terminal state, for example, completed.

Note: Business Console displays only valid actions for the process step. When the current status of the process step does not permit any actions, no valid actions are displayed in the drop-down menu.

Cancelling a Process Step

You can cancel a process step and prohibit any downstream steps from executing. Other tracks in the instance will continue to execute normally.

To cancel a process step:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Cancel**.

> **Note:** You can also click a step on the process diagram and click **Cancel**.

4. Select the step or steps you want to cancel.
5. Click **Cancel**.
6. Click **Yes**.

Restarting a Process Step

You can restart a failed process step and edit its pipeline data.

To restart a process step:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Restart**.

> **Note:** You can also click a step on the process diagram and click **Restart**.

4. Select the step you want to restart and optionally edit the pipeline data for the step.
5. Click **Restart**.
6. Click **Yes**.

Skipping a Process Step

You can edit the pipeline data and skip a process step, which will stop execution of this step and continue to the next downstream step.

To skip a process step:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Skip**.

> **Note:** You can also click a step on the process diagram and click **Skip**.

4. Select the step or steps you want to skip.
5. Click **Skip**.
6. Click **Yes**.

Playing a Paused Process Step

You can edit the pipeline data and play a paused step to resume the execution of this step.

To play a paused process step:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Play Paused**.

   Note: You can also click a step on the process diagram and click **Play**.

4. From the drop-down menu, select the paused step to play.
5. Click **Play**.

Playing a Process Step

You can play step, which is not yet executed, which creates a new track that runs simultaneously with the original track(s) in the process instance. Playing a step allows you to select from the available step pipeline data to start this new track.

To play a process step:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Play**.
4. From the drop-down menu, select the step to play.
5. Click **Edit Input**.
6. From the drop-down menu, select the pipeline input data to be used for the played step.
7. Click **Play**.

Injecting a Process

You can dynamically inject processes in a non-terminal process instance containing a paused or failed process step. When injecting a process from a paused step, you have the following options:

- Continue - after injecting the process, the process instance continues execution.
- Pause on return - the process instance remains with Needs-Attention status, and the step remains with Paused status, for further user interaction.

Note: Multiple process injections are allowed.

To inject a process:

1. In Business Console, on the **Processes** tab, select a process instance.
2. In the process instance details window, click the **Actions** drop-down menu.
3. Click **Inject process**.

   Note: You can also click a paused or failed step on the step diagram and click **Inject process**.

4. Select the process step to inject process.
5. Click **Select process**.
6. Select the process to inject.
7. When injecting from a paused step, select the **Play paused step after injecting process?** checkbox if you want to resume the paused step after the injected process returns control to this process instance.

   Note: This option is not available on failed step.

8. Click **Edit Input**.
9. From the drop-down menu, select the data to use with process injection.
10. Click **Inject**.

Setting or Removing Breakpoints for Process Steps

You can set breakpoint on one or more process steps to intervene in the process before the step executes. You can set and remove breakpoints for steps of a non-terminal process instance at runtime in the Business Console user interface.

Breakpoints apply only when the step is pending execution. Setting breakpoints on steps that are already completed does not have any effect.

Note: Breakpoints cause execution to pause before the step actually runs, not after.

To set or remove a breakpoint on a process step

1. In Business Console, on the Processes tab, click a process instance with status Running or Needs-Attention.
2. In the Process Diagram panel, select the step.
3. From the actions menu, select **Set breakpoint** or **Remove breakpoint**.

When the process instance execution reaches a breakpoint, the corresponding step is not executed and the step status is set to **Paused**. The status of the process instance is set to **Needs-Attention**.

Process Step Statuses

The following table shows the valid statuses for a step in a process instance.

Status	Description	Available Actions
Running	The step is running.	Cancel, Skip
Completed	The step was completed normally.	None
Failed	The step returned an exception and did not complete normally. The status of the process instance that holds this step is changed to Needs-Attention. Dynamic process injection is available for a failed step.	Restart, Inject Process, Cancel, Skip
Paused	The step execution reached a breakpoint. The status of the process instance that holds this step is changed to Needs-Attention. Dynamic process injection is available for a paused step.	Play, Inject Process, Cancel, Skip
Interrupted	The step activity was interrupted by a boundary event with an interrupting property.	None
Skipped	The step execution was skipped.	None
Waiting	The step is waiting for additional sequence flow or inbound events. A step with such status is not considered running or complete and all step actions are valid for it.	Cancel, Skip, Play
Redirected	A breakpoint was redirected to a dynamic step. The step is waiting for the injected process to complete execution. The step status automatically returns to **Paused** or **Running** depending on the return action that you select with the dynamic injection activity.	Cancel, Skip

Process Model Inflight Version Upgrade

With Dynamic Business Orchestrator process models, you are able to upgrade the version of the process models inflight. To upgrade the version of a process model, the model must be operationally running, which means that the model has to be in one of the following statuses: Running, Suspended, Needs-Attention. Upgrading to a new model version is done in webMethods Monitor and affects the process models, not individual process instances. Upgrading the process model version is done in the **Model Detail** page or in the **Business Processes** page. After choosing **Execution Enabled**, you can activate the new process model version. If there are process models of a previous version that are operationally running, you are prompted to upgrade model versions. This gives you the following choices:

- Clicking **YES** results in all running models with previous versions to be upgraded to the model version you are activating. Any models that are already completed are not upgraded.
- Clicking **NO** results in no version change for already running processes and they continue to execute with their initial versions. All new process instances will start with the newly activated process model version.

When the version of a process model has been upgraded, the information for the model version and name of the model will be updated in the Dynamic Business Orchestrator user interface for every affected process instance. Process models show in the **Process details** header any previously run versions.

Note: Even if any steps are removed with the upgraded model version, all steps that have already run are listed in the step summary table and the step name is matched to the last uploaded version of the model that has this step. Any steps that are removed with process version upgrade are only listed in the step summary table and not shown on the Process Diagram.

Dynamic Process Injection in Dynamic Business Orchestrator

Dynamic Business Orchestrator can dynamically execute process models by using dynamic process injection.

Dynamic process Injection enables you to redirect the process execution to a dynamic call activity and return to the original point of redirection.

When you use dynamic process injection, the status of a paused step is changed to **Redirected** and the process instance status is changed to **Running** while the call activity is executing. When the call activity completes, the resulting pipeline is merged at the original paused step. At this point, there are two possibilities for the original paused step:

- The step status is changed to **Paused** and the process instance status is changed to **Needs-Attention**.
- The step automatically executes with the updated pipeline. The step status is changed to **Running** and the process instance status is changed to **Running**.

Injecting a process from a failed step does not allow step execution to automatically continue on return. A failed step remains in failed state and the process instance status is changed to **Needs-Attention**.

Setting Design-Time Breakpoints for Process Models

Breakpoints can be added to steps in a process model at design-time in Software AG Designer. When a breakpoint is added to a specific step in the process model, the process instance execution will result in a pause at that specific step for every process instance of the process model. Breakpoints set in a process model in Designer can only be removed from the process model in Designer.

To add a breakpoint to a step in a process model

1. Edit your process model in the Dynamic Process Development perspective in Designer.
2. Select a step to add a breakpoint to.
3. Go to **Properties** > **General** and select the **Breakpoint** checkbox.
4. Save your changes.
5. Upload the process model.

Path Forecasting for a Process Instance

About Path Forecasting

You can check the path forecasting for a process instance in the Processes tab of webMethods Business Console. Path forecasting is based on aggregated historical data collected by Optimize and is available for currently running process instances that have been enabled for analysis.

When viewing the details for a process instance, you can select a forecast path and view the following estimated data for that path:

- **Estimated Completion Time** - The estimated time of completion if the forecast path is taken.
- **Percentage Complete** - The estimated percentage of completion for the process instance based on the selected forecast path's Average Path Cycle Time.

- **Average Path Cycle Time** - The average duration of the forecast path, calculated based on aggregated average step duration of the forecast path.
- **Average Process Cycle Time** - The average execution duration of previously completed process instances. Process instances that were not fully completed do not contribute to the average cycle time.
- **Path Frequency** - The frequency of the forecast path taken based on samples of historical data.

The estimated time of completion data is displayed for the entire process instance, not just for a single step. As the number of previously completed process instances increases, the accuracy of estimation also improves, because the estimation is based on a larger historical sample.

The path forecasting feature uses Optimize to provide estimations based on previously completed process instances. Optimize has a mode for calculating process and step statistical metrics, which is governed by the Optimize Analytic Engine's Monitor Behavior Setting. This setting has three modes of operation:

- All - all days of the week contribute to the same average.
- Work - weekdays contribute to one average, while weekend days contribute to a separate average.
- Day - each day of the week has its own average.

For more information on specifying statistical intervals, see *Administering webMethods Optimize*.

You should take into account the Optimize statistical mode of operation when checking estimated data for a forecast path. For example, if today is Tuesday and the Optimize statistical mode is "all days are different", the estimation is based on past process instances completed on a Tuesday.

Note: The Optimize Analytic Engine only calculates the process and step metric after the end of the day and averages do not include process instances for the current day.

Configuring Your System for Process Instance Path Forecasting

Configuring your system for process instance path forecasting is also necessary for stage timeline to show data. Stage data is also collected by Optimize.

To configure your system to use path forecasting for process instances:

1. In: **My webMethods** > **System Settings** > **Servers**, select one of the following server environments:
   - BAM
   - BPM and BAM

2. In **Business** > **Business Processes** open your process for editing and on the Process Details tab select the **Analysis Enabled** check box to enable the process for analysis.
3. Configure Optimize in My webMethods. For information on configuring Optimize, see the Optimize documentation.
4. Go to webMethods Business Console > **Administer** Business Console.
5. Define the **Analytical Engine Settings** listed in the following table:

Setting	Description
Analytical Engine URL	The URL of Analytic Engine.
Analytical Engine Username	User credentials for Analytic Engine.
Analytical Engine User Password	Password for Analytic Engine.

Viewing Estimated Data for a Forecast Path

To view the estimated data forecast path of a process instance:

1. In webMethods Business Console Processes tab.
2. Click a running instance to display the process instance details window.
3. In the **Process Diagram** panel, enable **Path Forecasting** by clicking the **On** radio button.

   Business Console displays the **Path Forecasting** bar to the left of the process diagram.
4. Click one of the dots on **Path Forecasting** bar to see a forecast path for the process instance.

   The different forecast paths are sorted from most common to least common by default. You can change the type of sorting from the drop-down list.
5. On the Path Information window, click **Show Stats** to view the estimated data for the forecast path.

When you select a forecast path, the path is highlighted. This shows whether the different parts of the path are completed (blue highlight) or not completed (black highlight). Forecast paths are always sequential and parallel paths are not taken into account.

Viewing Process Instance Alerts and Error Notifications

You can view the process instance alerts and error notifications for the last 24 hours in the Processes tab of webMethods Business Console.Alerts and notifications older than 24 hours are displayed separately in a different list.

Note: Only 5 notifications are displayed at a time. Clicking **View More** shows the next 5 notifications.

IX Dynamic Business Orchestrator Built-In Services

Dynamic Business Orchestrator Built-In Services Location

The built-in services in this chapter are installed on Integration Server as part of the WmDBO package.

The services and supporting elements are located in the \pub folder and you can access them from the **Package Navigator** view in Designer . You can use these services as templates to create services in Designer that perform a wide variety of actions on the services running in Dynamic Business Orchestrator on the connected Integration Server.

For additional information about working with services in Designer, see *webMethods Service Development Help*.

Elements in the WmDBO\pub.dbo.instance.control Folder

CustomId

WmDBO. Specification that describes the inputs and outputs necessary for a custom ID service.

Location in Package Navigator

pub.dbo.instance.control:CustomId

Input Parameters

ProjectName	**String** The name of the project that contains the process model.
ModelName	**String** The name of the model for which you wish to create a custom ID.
StepId	**String** ID of the step.

Output Parameters

None.

cancel

WmDBO. This service cancels a specified process instance.

Location in Package Navigator

pub.dbo.instance.control:cancel

Input Parameters

InstanceId	**String** Process instance ID of the process instance you want to cancel.
UserName	**String** Optional. Username for the user that is cancelling the process instance.

Output Parameters

None.

getStepInput

WmDBO. This service retrieves the input data used when the step is executed. The output of this service can be used when restarting a failed step.

Location in Package Navigator

pub.dbo.instance.control:getStepInput

Input Parameters

InstanceId	**String** The instance ID of the process instance containing the step with the desired input.
StepId	**String** ID of the step.

Output Parameters

StepInput	**String** The retrieved input data of the step.

logCustomId

WmDBO. This service associates a "friendly name" (the customID) with a Process Instance. This friendly name can be used to search for the process instance in webMethods Monitor.

This service is used within a process instance and affects the currently executing instance.

Location in Package Navigator

pub.dbo.instance.control:logCustomId

Input Parameters

CustomId	**String** The "friendly name" you want to assign to a Process Instance.

Note: The use of the characters “&” and “=”are restricted in this parameter. For example, if you create a custom ID with a format of `<fieldname1>=<valuename1>` or `<fieldname1>=<valuename1>&<fieldname2>=<valuename2>`,

then Monitor will create a column for each field name and display the value of the value name in that column.

Output Parameters

None.

logStepMessage

WmDBO. This service is used within a process instance to log step activity messages and affects the currently executing process instance and step.

Location in Package Navigator

pub.dbo.instance.control:logStepMessages

Input Parameters

BriefMessage	**String** Shortened version of the full message. The message can be up to 240 bytes.
FullMessage	**String** Optional. Complete message. The message can be up to 1024 bytes.
MessageType	**String** Flag indicating the type of message. The following values apply: ■ `MESSAGE` — Indicates that the message is informational and no action is needed. ■ `WARNING` — Indicates that the message is a warning message. ■ `ERROR` — Default. Indicates that the message is an error message.

Output Parameters

None.

restartAllFailedSteps

WmDBO. This service restarts all failed steps in a specified process instance.

Location in Package Navigator

pub.dbo.instance.control:restartAllFailedSteps

Input Parameters

InstanceId	**String** Process instance ID of the process instance for which you want to restart all failed steps.
UserName	**String** Optional. Username for the user that is restarting the failed steps within the process instance.

Output Parameters

None.

restartFailedStep

WmDBO. This service restarts a specific failed step in a process instance.

Location in Package Navigator

pub.dbo.instance.control:restartFailedStep

Input Parameters

InstanceId	**String** Process instance ID of the process instance that contains the failed step you want to restart.
StepId	**String** The step ID of the step you want to restart.
StepInput	**String** The input data used to restart the failed step. If not specified, the input that was used previously will be used for the restart. You can get the previous step input using the `pub.dbo.control.instance:getStepInput` service, then modify that data as necessary.
UserName	**String** Optional. Username for the user that is restarting the failed step.

Output Parameters

None.

resume

WmDBO. Service that allows users to resume processing for a previously suspended process instance.

Location in Package Navigator

pub.dbo.instance.control:resume

Input Parameters

InstanceId	**String** This is the instance ID for the instance you want to resume.
UserName	**String** Optional. Username for the user that is resuming the process instance.

Output Parameters

None.

suspend

WmDBO. Service that allows users to suspend a process instance.

Location in Package Navigator

pub.dbo.instance.control:suspend

Input Parameters

InstanceId	**String** This is the instance ID for the instance you want to suspend.
UserName	**String** Optional. Username for the user that is suspending the process instance.

Output Parameters

None.

changeVersion

WmDBO. This service changes the version of a process instance.

Location in Package Navigator

pub.dbo.instance.control:changeVersion

Input Parameters

InstanceId	**String** Process instance ID for the instance of which you want to change the version.
ToModelVersion	**String** The model version to which you want to change the process instance.
UserName	**String** Optional. Username for the user that is changing the process instance version.

Output Parameters

None.

complete

WmDBO. This service forces the completion of a specified process instance.

Location in Package Navigator

pub.dbo.instance.control:complete

Input Parameters

InstanceId	**String** This is the instance ID for the instance you want to suspend.
StepInput	**IData** Optional. The pipeline to be used when completing the process instance.
UserName	**String** Optional. Username for the user that is suspending the process instance.

Output Parameters

None.

throwStepHandledException

WmDBO. This service is used to override an unhandled exception behavior for a step. The exception is then treated as a handled exception.

Location in Package Navigator

pub.dbo.instance.control:throwStepHandledException

Input Parameters

ErrorMessage	**String** The error message text to be displayed when the exception is thrown.

Output Parameters

None.

throwStepUnhandledException

WmDBO. This service is used to override a handled exception behavior for a step. The exception is then treated as an unhandled exception.

Location in Package Navigator

pub.dbo.instance.control:throwStepUnhandledException

Input Parameters

ErrorMessage	**String** The error message text to be displayed when the exception is thrown.

Output Parameters

None.

gotoCallActivity

WmDBO. This service dynamically invokes a Call Activity at a step that has been paused for execution either via a breakpoint or as a result of a failed step execution.

Location in Package Navigator

pub.dbo.instance.dynamic:gotoCallActivity

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
GotoStepId	**String** The step ID of the dynamic Call Activity step in your process model.
StepInput	**IData** Optional. The pipeline to be used when the Call Activity is dynamically executed.
ProjectName	**String** The name of the project that contains your process model.
ModelName	**String** The name of your process model.
ReturnStepId	**String** The step ID of your process model that is paused for execution.
ContinueOnReturn	**Boolean** When the value is set to `true`, instructs Dynamic Business Orchestrator to automatically continue execution of the step that is paused for execution after the dynamic Call Activity returns. When the value is set to `false`, the step will remain paused for execution.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

skipStep

WmDBO. This service skips the execution of the specified step. The step must be in a running, failed, or paused state. The step is marked as Skipped and the sequence flow from the step is executed normally.

Location in Package Navigator

pub.dbo.instance.dynamic:skipStep

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
StepId	**String** The step ID of the step to skip.
StepInput	**IData** Optional. The pipeline to be used when the Call Activity is dynamically executed.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

playStep

WmDBO. This service creates a new execution path at the specified step. The specified step cannot currently be running. This service essentially creates a new execution path in the process model without altering the existing execution paths.

> This service could result in the process instance not moving to a Completed state as expected.

Location in Package Navigator

pub.dbo.instance.dynamic:playStep

Input Parameters

InstanceId	**String** Process instance ID of the process instance.

StepId	**String** The step ID of the step to play.
StepInput	**IData** Optional. The pipeline to be used when the step is executed.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

playAllPausedSteps

WmDBO. This service plays all steps that have been paused for execution for the specified process instance.

Location in Package Navigator

pub.dbo.instance.dynamic:playAllPausedSteps

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

playPausedStep

WmDBO. This service plays a single step that has been paused for execution for the specified process instance.

Location in Package Navigator

pub.dbo.instance.dynamic:playPausedStep

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
StepId	**String** The step ID of the paused step.
StepInput	**IData** Optional. The pipeline to be used when playing the paused step.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

cancelStep

WmDBO. This service cancels the execution of the specified step. The step must be in a running, failed, or paused state. The step is marked as Interrupted and no sequence flow is issued from the cancelled step.

Location in Package Navigator

pub.dbo.instance.dynamic:cancelStep

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
StepId	**String** The step ID of the step to cancel.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

getBreakPoints

WmDBO. This service returns the list of steps for which there are breakpoints set for a given process instance.

Location in Package Navigator

pub.dbo.instance.dynamic:getBreakPoints

Input Parameters

InstanceId	**String** Process instance ID of the process instance.

Output Parameters

BreakPoints	**String List** String list of process step IDs that have a breakpoint set.

removeBreakPoint

WmDBO. This service removes the break point that has been set at a particular step ID in a process instance.

Location in Package Navigator

pub.dbo.instance.dynamic:removeBreakPoint

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
StepId	**String** The step ID of the paused step.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

setBreakPoint

WmDBO. This service sets a break point at a particular step ID in a process instance. The step will pause for execution when the break point is encountered.

Location in Package Navigator

pub.dbo.instance.dynamic:setBreakPoint

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
StepId	**String** The step ID of the paused step.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

Correlation

WmDBO. Specification that describes the inputs and outputs required for a correlation service.

Location in Package Navigator

pub.dbo.instance.correlation:Correlation

Input Parameters

ProjectName **String** The name of the project that contains your model.

ModelName **String** The name of the model for which you are establishing correlation.

ModelVersion **String** Version of the process model with which this invocation of the correlation service is involved.

> **Note:** Because a single correlation service can be associated with steps from more than one process model version, you can use the *ModelID* and *ModelVersion* to identify the process model version using the correlation service at run time.

StepId **String** ID of the step in the process model version with which this invocation of the correlation service is involved (for example, `N3`).

> **Note:** Because a single correlation service can be associated with multiple steps in a process model version, you can use *StepID* to identify the specific step at run time.

DocumentName **String** Name of the document (for example, "OrderDocument").

DocumentType **String** Name of the document type (for example, "orders.sap:OrderDocument").

Output Parameters

CorrelationId **String** An abstract ID that correlates to the actual process instance ID of the running process. For example:

"CUSTOMER-0003456977::ORDER-19477593-AR9-1000". All documents bound for the same instance of the process must return the same correlation ID. Similarly, correlation IDs must be unique across all process instances.

create

WmDBO. This service is used within a process instance to create a correlation ID and affects the currently executing process instance.

Location in Package Navigator

pub.dbo.instance.correlation:create

Input Parameters

CorrelationId	**String** An abstract ID that correlates to the actual process instance ID of the running process. For example: "CUSTOMER-0003456977::ORDER-19477593-AR9-1000". All documents bound for the same instance of the process must return the same correlation ID. Similarly, correlation IDs must be unique across all process instances.

Output Parameters

None.

delete

WmDBO. This service is used within a process instance to delete a correlation ID and affects the currently executing process instance.

Location in Package Navigator

pub.dbo.instance.correlation:delete

Input Parameters

CorrelationId	**String** The correlation ID to delete for this process instance.

Output Parameters

None.

lookup

WmDBO. This service is used to look up the InstanceId associated with the specified CorrelationId. If the CorrelationId does not exist, an exception is thrown.

Location in Package Navigator

pub.dbo.instance.correlation:lookup

Input Parameters

CorrelationId	**String** CorrelationId to lookup.

Output Parameters

InstanceId	**String** The InstanceId associated with the CorrelationId.

Elements in the WmDBO\pub.dbo.model.control Folder

joinProcess

WmDBO. This service is used to join(correlate into) a running process instance. This is done by specifying the service transport on the message event in Designer.

Location in Package Navigator

pub.dbo.model.control:joinProcess

Input Parameters

ProjectName	**String** The name of the project that contains the model to join.
ModelName	**String** The name of the model to join.
DocumentType	**String** The document type of the document being sent into the process instance. This corresponds to the doc type specified on the message event in the model.
DocumentData	**IData document** The document that contains the data specified in the document type.

Output Parameters

InstanceId	**String** The instance ID of the joined process instance.

restartFailedSteps

WmDBO. This service restarts all failed steps in all process instances of a specified process model.

Location in Package Navigator

pub.dbo.model.control:restartFailedSteps

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model.

UserName	**String** Optional. Username for the user that is restarting the failed steps within the process instances of the specified model.

Output Parameters

None.

resume

WmDBO. This service is used to resume processing of all suspended instances of a process model.

Location in Package Navigator

pub.dbo.model.control:resume

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model you want to resume.
UserName	**String** Optional. User that is resuming the process model.

Output Parameters

None.

startProcess

WmDBO. This service is used to start a process instance. This is done by specifying the service transport on the message event in Designer.

Location in Package Navigator

pub.dbo.model.control:startProcess

Input Parameters

ProjectName	**String** The name of the project that contains the model to join.
ModelName	**String** The name of the model to join.

DocumentType	**String** The document type of the document being sent into the process instance. This corresponds to the doc type specified on the message event in the model.
DocumentData	**IData document** The document that contains the data specified in the document type.
CallbackService	**String** Optional. The name of the service to be invoked when the process instance reaches a terminal state.

Output Parameters

InstanceId	**String** The instance ID of the started process instance.

changeVersion

WmDBO. This service changes the version of all process instances for a given model from one version to another.

Location in Package Navigator

pub.dbo.model.control:changeVersion

Input Parameters

ProjectName	**String** The name of the project that contains the model to change the version of.
ModelName	**String** The name of the process model to change the version of.
FromModelVersion	**String** Optional. The model version of the process instances you want to change. **Important:** When this parameter is not set, all process instances of the model type are changed to the model version specified in the *ToModelVersion* parameter regardless of the model version they are currently executing.
ToModelVersion	**String** The model version to which you want to change the process instances.
UserName	**String** Optional. Username for the user that is changing the process instances version.

Output Parameters

None.

suspend

WmDBO. This service is used to suspend processing of all instances of a process model.

Location in Package Navigator

pub.dbo.model.control:suspend

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model you want to suspend.
UserName	**String** Optional. User that is suspending the process model.

Output Parameters

None.

Elements in the WmDBO\pub.dbo.model.admin Folder

activate

WmDBO. This service is used to set the active model version.

Location in Package Navigator

pub.dbo.model.admin:activate

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model you want to activate.
ModelVersion	**String** Version of the process model to activate.

Output Parameters

None.

deactivate

WmDBO. This service is used to disable all versions of a process model.

Location in Package Navigator

pub.dbo.model.admin:deactivate

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model you want to disable.

Output Parameters

None.

getActiveVersion

WmDBO. This service is used to get the active version of a process model.

Location in Package Navigator

pub.dbo.model.admin:getActiveVersion

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model you want to get properties for.

Output Parameters

ModelVersion	**String** The active version of the process model.

list

WmDBO. This service is used to list all model versions for a process model and their properties.

Location in Package Navigator

pub.dbo.model.admin:list

Input Parameters

None.

Output Parameters

Models	**Document List** List of all versions of the process model and their properties. Each entry in the list contains *ProjectName* , *ModelName* , *ModelVersion* , *ModelDisplayName* , *ModelType* and *Active* .

validate

WmDBO. This service is used to validate a process model.

Location in Package Navigator

pub.dbo.model.admin:validate

Input Parameters

ProjectName	**String** The name of the project that holds the process model.
ModelName	**String** The name of the process model to be validated.
ModelVersion	**String** The version of the process model to be validated.

Output Parameters

Design Issues	**String** The list of design validation issues.
Runtime Issues	**String** The list of run-time validation issues.

playAllPausedSteps

WmDBO. This service plays all steps that have been paused for execution for all process instances for the specified model.

Location in Package Navigator

pub.dbo.model.dynamic:playAllPausedSteps

Input Parameters

InstanceId	**String** Process instance ID of the process instance.
UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

Elements in the WmDBO\pub.dbo.system.control Folder

disable

WmDBO. This service is used to disable the runtime so that no further processing occurs.

Location in Package Navigator

pub.dbo.system.control:disable

Input Parameters

DisabledReason	**String** Text that is displayed as the reason why the runtime is disabled.

Output Parameters

None.

enable

WmDBO. This service is used to enable the runtime after it has been disabled.

Location in Package Navigator

pub.dbo.system.control:enable

Input Parameters

None.

Output Parameters

None.

isEnabled

WmDBO. This service is used to check whether the runtime is currently disabled.

Location in Package Navigator

pub.dbo.system.control:isEnabled

Input Parameters

None.

Output Parameters

IsEnabled	**String** Displays `true` if the runtime is enabled and `false` if the runtime is disabled.
DisabledReason	**String** Displays the reason why the runtime is disabled.

restartFailedSteps

WmDBO. This service restarts all failed steps for all process models.

Location in Package Navigator

pub.dbo.system.control:restartFailedSteps

Input Parameters

UserName	**String** Optional. Username for the user that is restarting the failed steps within the process instances of all process models.

Output Parameters

None.

restartAllFailedSteps

WmDBO. This service restarts all failed steps for all process instances.

Location in Package Navigator

pub.dbo.system.control:restartAllFailedSteps

Input Parameters

UserName	**String** Optional. Username for the user that is restarting the failed steps within the process instances of all process models.

Output Parameters

None.

playAllPausedSteps

WmDBO. This service plays all steps that have been paused for execution for all process instances.

Location in Package Navigator

pub.dbo.system.dynamic:playAllPausedSteps

Input Parameters

UserName	**String** Optional. Username of the user requesting this operation.

Output Parameters

None.

Elements in the WmDBO\pub.dbo.provision Folder

add

WmDBO. This service adds a model to the provision list. A model on the provision list instructs Dynamic Business Orchestrator to load all versions of that model for execution.

Location in Package Navigator

pub.dbo.provision:add

Input Parameters

ProjectName	**String** The name of the project that contains the process model.
ModelName	**String** The name of the process model to provision for execution.

Output Parameters

None.

list

WmDBO. This service returns the list of models that are provisioned for the current Dynamic Business Orchestrator node.

Location in Package Navigator

pub.dbo.provision:list

Input Parameters

None.

Output Parameters

Models	**Document List** List of all provisioned process models. Each entry in the list contains *ProjectName* and *ModelName* .

refresh

WmDBO. This service instructs the current Dynamic Business Orchestrator node to re-initialize based on the contents of the provision file. This is functionally equivalent to reloading the WmDBO package.

Location in Package Navigator

pub.dbo.provision:refresh

Input Parameters

None.

Output Parameters

None.

remove

WmDBO. This service removes a model type from the provision list for the current Dynamic Business Orchestrator node.

Location in Package Navigator

pub.dbo.provision:remove

Input Parameters

ProjectName	**String** The name of the project that contains the process model.
ModelName	**String** The name of the process model to remove.

Output Parameters

None.

Elements in the WmDBO\pub.dbo.repository Folder

list

WmDBO. This service returns the list of process models that are stored in the Dynamic Business Orchestrator central repository.

Location in Package Navigator

pub.dbo.repository:list

Input Parameters

None.

Output Parameters

Models	**Document List** List of all versions of process models. Each entry in the list contains *ProjectName* , *ModelName* , *ModelVersion* , *ModelDisplayName* , *UpdateTimestamp* , and *Active* .

read

WmDBO. This service returns the model definition from the Dynamic Business Orchestrator central repository for the specified model version.

Location in Package Navigator

pub.dbo.repository:read

Input Parameters

ProjectName	**String** The name of the project that contains the process model.
ModelName	**String** The name of the process model to read.
ModelVersion	**String** The version of the process model to read.

Output Parameters

ModelDef	**Object** An IDATA encoded representation of the ModelDef object.

ModelMap **Object** An IDATA encoded representation of the ModelMap object.

UpdateTimestamp **Date** The date the model was last updated in the repository.

remove

WmDBO. This service removes the model definition from the Dynamic Business Orchestrator central repository for a specified model version.

Location in Package Navigator

pub.dbo.repository:remove

Input Parameters

ProjectName **String** The name of the project that contains the process model.

ModelName **String** The name of the process model to remove.

ModelVersion **String** The version of the process model to remove.

Output Parameters

None.

removeAllVersions

WmDBO. This service removes the model definition from the Dynamic Business Orchestrator central repository for all versions of the specified model.

Location in Package Navigator

pub.dbo.repository:removeAllVersions

Input Parameters

ProjectName **String** The name of the project that contains the process model.

ModelName **String** The name of the process model to remove.

Output Parameters

None.

save

WmDBO. This service updates the model definition in the Dynamic Business Orchestrator central repository for the specified model version.

Location in Package Navigator

pub.dbo.repository:save

Input Parameters

ModelDef	**Object** An IDATA encoded representation of the ModelDef object.
ModelMap	**Object** An IDATA encoded representation of the ModelMap object.

Output Parameters

None.