webMethods Dynamic Business Orchestrator Help

Innovation Release

Version 10.0

June 2017

WEBMETHODS

This document applies to webMethods Dynamic Business Orchestrator Version 10.0 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-100-20170616

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 storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at "http://documentation.softwareag.com". The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

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".

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 following values:

Field	Properties
Action Type	Create
Action Component	Archive
Version	Latest

   b. In the **Connection** section, define the connection to your Archive database.

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.

Field	Properties
Admin ID	Add the Archive database administrator.

Field	Properties
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
`ISCORE_SCHEMA`	To archive data from the IS Core Audit Log database, specify the following:

Parameter	Entry
	■ **Oracle**: IS Core Audit Log database user ■ **SQL Server**: IS Core Audit Log database name ■ **DB2**: IS Core Audit Log schema name

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.

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**

To allow users to...	In the Access Privileges section, select the check box...
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...
Stop, suspend, and resume process instances.	**Business Monitoring > Processes > Stop, Suspend, Resume**
Resubmit process instances.	**Business Monitoring > Resubmit**
Modify the pipeline for a process instance and resubmit the process instance.	**Business Monitoring > Modify and Resubmit**
Resubmit services.	**Integration Monitoring > Services > Resubmit**
Modify the pipeline for a service and resubmit that service.	**Integration Monitoring > Services > Modify and Resubmit**
Resubmit documents.	**Integration Monitoring > Documents > Resubmit**
Modify and resubmit documents.	**Integration Monitoring > Documents > Modify and Resubmit**

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**
Dynamically inject processes	**Business Monitoring > Processes > DBO Process Injection**

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 24.
- 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 24.

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**.

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:

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 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. In the **Save As** window, specify the following fields:

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 all mapping information is saved to an XML file.

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 **Edit Data Mapping**, or right-click the step and select **Edit Input Data Mapping** or **Edit Output Data Mapping**.
4. Edit the data mapping in the **Pipeline 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.

5. Click **Save**.

Specific Mapping Information for Different Types of Steps

Different types of process steps have specific behavior in regard of data mapping.

Start Message Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
The ProcessInfo document	All outputs specified in the Outputs table from the **Properties** > **Inputs/ Outputs** tab.

On the left side	On the right side
The trigger document, that is the document from the **Properties** > **Implementation** tab.	

Start None Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
The ProcessInfo document	All outputs specified in the Outputs table from the **Properties** > **Inputs/ Outputs** tab.
	Any documents listed in the **Properties** > **Global Process Specification** tab for the process.

Catch Message Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
The caught document, that is the document from the **Properties** > **Implementation** tab.	The caught document, that is the document from the **Properties** > **Implementation** tab.
The ProcessInfo document	All outputs specified in the Outputs table from the **Properties** > **Inputs/ Outputs** tab.

Boundary Message Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
The caught document, that is the document from the **Properties** > **Implementation** tab.	The caught document, that is the document from the **Properties** > **Implementation** tab.
The ProcessInfo document	All outputs specified in the Outputs table from the **Properties** > **Inputs/ Outputs** tab.

Throw Message Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
All inputs specified in the Inputs table from the **Properties** > **Inputs/Outputs** tab.	The document to be thrown, that is the document from the **Properties** > **Implementation** tab.
The ProcessInfo document	

End Message Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
All inputs specified in the Inputs table from the **Properties** > **Inputs/Outputs** tab.	The document to be thrown, that is the document from the **Properties** > **Implementation** tab.
The ProcessInfo document	

End None Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
All inputs specified in the Inputs table from the **Properties** > **Inputs/Outputs** tab.	None.
The ProcessInfo document	You can manually add elements to this side and map to them.

If the step belongs to a child process, then the mapped elements can be returned to the parent.

End Error Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
All inputs specified in the Inputs table from the **Properties** > **Inputs/Outputs** tab.	None.
The ProcessInfo document	

End Terminate Event Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
All inputs specified in the Inputs table from the **Properties** > **Inputs/Outputs** tab.	None.
The ProcessInfo document	

Boundary Error Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
None.	All outputs specified in the Outputs table from the **Properties** > **Inputs/ Outputs** tab.
You can manually add elements to this side and map from them. **Note:** Manually added elements are not contained in the pipeline and do not contain any run time value unless manually set.	The ProcessInfo document

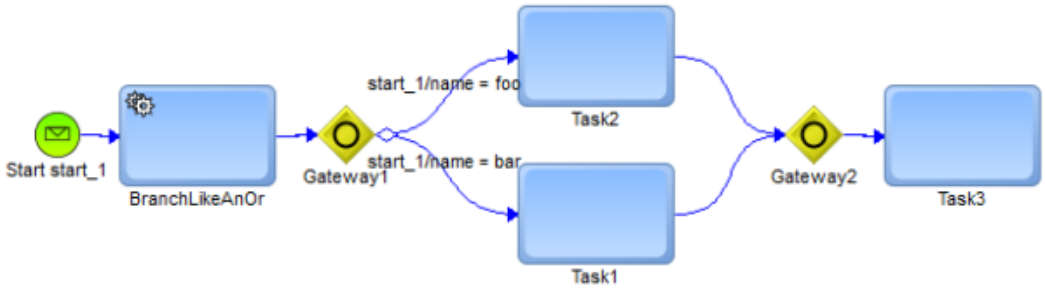
Activity Steps

When editing the data mapping for this step type, the **Pipeline view** displays the following:

On the left side	On the right side
All inputs specified in the Inputs table from the **Properties** > **Inputs/Outputs** tab.	None.
The ProcessInfo document	You can manually add elements to this side and map from them, but those mappings do not affect the manual task in any way.

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).
- **Business Calendar Value**- specify the timeout value using a business calendar inMy webMethods Server. Select the business calendar and specify a timeout value in the following format: days, hours, minutes, and seconds.

4. Save the process.

Using Complex Join Expressions

Many of the available Designer step types support a complex join. The specific behavior of a complex join is defined by a join expression that you create. Evaluation of the join expression at run time determines whether the join is satisfied or unsatisfied.

You must consider the following before defining a join expression:

- If no join expression is defined, process generation will fail with an error stating that the join condition is invalid.
- A warning appears during generation if you do not use all of the available transitions in the condition.
- Designer does not perform any syntax validation. If you enter any unsupported characters in the expression editor, a run-time error will result.

Defining a Complex Join Expression

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

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.

To define a complex join expression

1. Open a process and click a step that has a complex join.
2. In the Properties view, click the **Implementation** page.
3. In the **Join Condition** area, 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.

Example

Consider a step that has three incoming transitions:

- Transition-from-Step-6(StepID:S6)
- Transition-from-Step-5(StepID:S5)
- Transition-from-Step-4(StepID:S4)

You want the join to be considered satisfied if any two of the three incoming transitions are true. The expression looks like this:

```
(Transition-from-S6 and Transition-from-S5) or (Transition-from-S4 and
Transition-from-S5) or (Transition-from-S6 and Transition-from-S4)
```

Synchronizing Process Runtime Settings with webMethods Monitor

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

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

In both cases, any changes to the stage runtime settings in a process model 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 runtime settings are always synchronized between the two locations.

In Designer, you must consider the following:

- When you open a process in Designer, it retrieves the runtime settings saved with the model in the local workspace.
- When you click the **Synchronize** button on the **Stages** page in the Properties view, or in the **Runtime Editable Properties** section of the **Runtime** page in the Properties view, Designer retrieves the runtime settings from the database and applies them to the process model. Saving the process model saves the settings to the local workspace.If you want your process model in Designer to display the current database runtime stage settings, you must click the **Synchronize** button on the **Stages** page of the Properties view.
- When you build and upload a process in Designer, the runtime settings in the model are saved in the Process Audit database, overwriting the settings stored in the database and applying the updated settings to the runtime environment. In addition, a warning message is recorded in the build and upload report. If you want your process model runtime settings in Designer to be written to the database, build and upload the process model. This overwrites the settings are stored in the database, and the setting are applied to the runtime environment.

To ensure that you work with the latest settings, click the appropriate **Synchronize** button immediately before you modify and save these settings.

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.

Synchronizing Stage Settings

As a best practice, you should ensure that your process model stage settings are always synchronized between Designer and Monitor.

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 build and upload the process model. Uploading the process overwrites the existing stage settings in the Process Audit database.

Using webMethods Business Rules

A webMethods Business Rule is a decision-making tree capable of complex behavior. Designing a rule is very straightforward. *Rule Entity* is the term used to describe the components. Specifically, those components are decision tables, rule sets, event rules, and rule actions. Each plays a specific role in the decision-making tree.

- Rule: a rule contains a decision table or a rule set with multiple tables.
- Decision table: contains rule actions (Data, Service, and Process).
- Rule action: indicates one of three types of actions to execute for a rule: Data actions, Service actions, and Process actions.
- Process action: indicates which action to execute for a process, for example: start, join, suspend, cancel, fail, resume and call a task.

Note that a process can invoke a rule, but a rule can also invoke a process.

For more on webMethods Business Rules and how they work, see *webMethods BPM Rules Development Help*.

Decision Entities

In webMethods Business Rules, a *decision entity* can refer to a decision table, a rule set, an event rule, or a rule action. Decision entities are used to make decisions about the ways in which rules are applied.

Decision Tables

Decision tables contain logic used to determine how a rule behaves. A rule may use a single decision table or multiple decision tables that work together. Decision tables can contain rule actions.

A decision table that contains a manual decision point (task action) is configured as follows:

- When creating the decision table, select the **Process Aware** check box to enable the decision table to work with processes.
- Map the TaskData.
- Match the inputs/outputs of the tasks to the decision table's Inputs/Outputs.

  Note: The decision table contains an extra parameter called ProcessData that is automatically generated when the decision table is created. This is mapped separately to the process, not to the task.

- Map the ProcessData of the decision table to the process.

  Important: Do not overwrite or drop ProcessData, because it contains necessary runtime data. ProcessData mappings must be only outgoing.

Rule Sets

When multiple decision tables work together, they form a rule set.

Event Rules

An event rule is a decision entity. It specifies the results triggered by an event that occurs during rule execution. If you use an event rule in a rule task, there must be a JMS trigger on the corresponding Integration Server.

Rule Actions

Rules can call a service and run that service: a service action. They can call a process to invoke (start), suspend, cancel, fail or if it is suspended, to resume itself. Rules can also call a user task within a process, so that a human can make a decision as to the way in which the rule is ultimately applied. This is a unique process action known as a *manual decision*, sometimes referred to as a task action.

Rule actions are constructed in the Rules Explorer view and then dragged and dropped into decision tables, which are used in rules.

Important: To use a rule action in a process, you must configure your **Minimum Logging Level** for the process to **5 - Process and all steps**. Designer uses this data to identify the requirements of the rule action.

Process Actions

A process action is a specialty rule action that affects an entire process. Actions you can apply to a process include the following:

- Start a new process instance
- Join a running process instance
- Invoke a user task (requiring a manual decision)

Manual Decisions

Manual decisions, sometimes referred to as task actions, are a special type of process action that uses a rule to instantiate a task. When the user completes the task, the Task Engine calls the Rules Engine with the result and the original call context.

Note: Manual decisions have nothing to do with manual tasks.

A decision table that contains a manual decision point (task action) is configured as described in Decision Tables.

For more information about working with rule actions and decision tables, see *webMethods BPM Rules Development Help*.

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.
Unknown	An input or output of unknown type.
Unknown list	A one-dimensional unknown array.

Specifying Referenced Process Start and Return Documents

When you configure a call activity step to statically invoke a webMethods referenced process, you must also specify start and return documents. Start documents are sent to the referenced process, and return documents are returned from the referenced process to the original process. Start and return documents are not needed when calling a BPMN callable process. A BPMN callable process requires a Global Process Specification in order to be called by a call activity step.

As with other steps, the inputs of a call activity step configured to invoke a referenced process can be any available pipeline process data from upstream in the process. Such

a call activity step must also publish the specified receive document to invoke another process.

If the call activity step input document(s) matches the selected start document(s) (and has values for all the required fields in the receive document), then no mapping is required after the process is built. If the input document does not match the start document, then you must map the data manually. Click the **Edit Data Mapping** links on the Inputs / Outputs page in the Properties view, or right-click a step and select **Edit Input Data Mapping** or **Edit Output Data Mapping**.

Similarly, the outputs of a call activity step can be any process data you choose to include. But if the outputs do not match the selected return document(s), they will require mapping after you build the process.

Available start documents are the sum of the subscription documents for all of the start event steps in the referenced process that can start new instances of the process. Available return documents are the sum of all the publication documents in the referenced process.

To specify referenced process start and return documents

1. Select a call activity step in the process editor. The step must be configured to statically invoke a webMethods reference process.
2. In the **Available Input Documents** section on the **Start / Return Documents** page in the Properties view, select the referenced process subscription documents that should be used to invoke the referenced process (CTRL + click to select multiple documents), and click **Add** to move them to the **Inputs** section.
3. In the **Available Output Documents** section on the **Start / Return Documents** page in the Properties view, select the referenced process publish documents you want returned to the parent process (CTRL + click to select multiple documents), and click **Add** to move them to the **Outputs** section.
4. Save the process.

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 43 and "Remove Inputs and Outputs" on page 44.
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 43 and "Remove Inputs and Outputs" on page 44.
5. Save the process.

Basic Process Properties

Properties Page	Property	Description
General	**Process Name**	The name of the process model. **Important:** If you change a process-wide property, such as renaming a process or a package, Designer prompts you to regenerate the process.
	Process ID	A system-generated process identifier. Not editable. The **Process ID** consists of the process project name, any intermediate folders, and the process name, each separated by a slash (/), if all characters are simple ASCII characters. If any non-ASCII characters are used, Designer uses an encoding scheme to render all characters in the **Process ID** in ASCII.

Properties Page	Property	Description
	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.
	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

Properties Page	Property	Description
		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. Adding, editing, and removing inputs and outputs for a global process is done the same way as it is for steps.

Invoking a User Task

To create an action to start a new user task 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 **Manual decision**.
4. Click **Next**.
5. On the Select Task page, select the task type you want to start with the action.
6. In the **Integration Server Name** list, select the Integration Server where the process is defined. Click **Next**.

7. 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**.
8. On the Process Action Return Value page, select a return value check box as required.
9. Click **Finish**.

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**.

Using webMethods Business Rules in Processes

To use a webMethods Business Rule in a process

1. In Designer, create a rule task on the process editor's canvas.
2. Configure the rule task to use a webMethods Business Rule.

> **Tip:** You can also drag and drop a webMethods Business Rule from the Rules Explorer view in the Rules perspective. For more information about the Rules perspective, see the *webMethods BPM Rules Development Help*.

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 54

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.

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.

V Administering Process Models

About Process Model Data Logging

Dynamic Business Orchestrator can log data for webMethods-executed process instances. You can view this data and perform actions on it in Monitor.

- For each process model version, you specify the amount and type of data to log in the **Logging Level** setting.
- When Monitor renders a process diagram, it shows all the possible paths that can be taken within a process instances. If you want to see the path the process instances actually took at run time, use the **Log Transitions** setting to enable process transition logging for the process model version. The lines for the path that was actually executed are displayed as heavier lines.
- You can select the **Diagnostic Logging** option to specify that log messages from instances started from a process model are logged to a separate file for diagnostic purposes. For more information, see "Enabling and Disabling Process Instance Diagnostic Logging" on page 61.

Important: When you regenerate a process model version, the logging settings return to the default values, and you must reset them if you want different settings.

About Process Model Logging Levels

The following table describes the available process model logging levels, with additional information about choosing a particular level. Further information about choosing a logging level is found in "Improving Process Logging Performance" on page 59.

You want to log...	In Monitor, you want to...			Set to...
	View process status?	View step status?	Restart a process?	
Nothing (that is, disable process logging)	No	No	No	**1 - None**
Process status when steps fail Input pipelines for failed steps Run-time values for document fields	At failed step	No	At failed step	**2 - Errors only**

You want to log...	In Monitor, you want to...			Set to...
	View process status?	View step status?	Restart a process?	
Process status Input pipelines for failed steps Run-time values for document fields Optionally, transitions	Yes	No	At failed step	**3 - Process only**
Process status and start step status Input pipelines for start steps and failed steps Run-time values for document fields Optionally, transitions	Yes	For start step	At start or failed step	**4 - Process and start events**
Process status and all step statuses Input pipelines for every step Run-time values for document fields Optionally, transitions Loop count and loop iteration status for all processed steps.	Yes	For all steps	At any step that has logged input pipeline	**5 - Process and all events, activities, and looped activities**

Improving Process Logging Performance

To improve process logging performance, consider the following:

- Select **2 - Errors only**, **3 - Process only**, or **4 - Process and start steps** as your logging level. Choose **5 - Process and all steps** only when you need ultimate quality of service.

- Store input pipelines only when absolutely necessary. It is usually sufficient to store pipelines for failed steps only. Remove all unnecessary data from pipelines to minimize the volume of data to store.
- For process steps that run services, there are two areas in which you could inadvertently log the same information twice:
  - Dynamic Business Orchestrator can write start and successful completion or failure log entries for process steps that run services. Services can write log entries that convey the same information.
  - Dynamic Business Orchestrator can store input pipelines for services that are run by process steps. Services can also log input pipelines.

  Coordinate your logging for these services to avoid logging the same information twice.

  Note: When coordinating logging, consider that when a service is run by a process step, that service is actually called by a wrapper service, making it a nested service (as opposed to a top-level service).

For instructions on setting up service logging, and for complete information on logging in general, see *webMethods Audit Logging Guide*.

Configuring Logging Settings for a Process Model Version

To configure the logging settings for a process model version:

1. In My webMethods, click **Navigate > Applications > Administration > Business > Business Processes**.
2. Find and edit the process model version.
3. In the Process Information Administration window, click the **Process Settings** tab, and do any or all of the following:
   - Select one of the available **Logging Level** values, as described in "About Process Model Logging Levels" on page 58.

     Important: A **Minimum Logging Level** setting is specified in Designer for each process model version. This setting in Designer controls the lowest logging level that you can set in Monitor.
   - Select the **Log Transitions** check box if you want to log process transitions for display in the process diagram. This requires a logging level that enables you to log transitions
   - Select the **Diagnostic Logging** check box if you want to log messages from a process instance to a separate log file for diagnostic purposes. For more information, see "Enabling and Disabling Process Instance Diagnostic Logging".
4. Click **Save**.

Enabling and Disabling Process Instance Diagnostic Logging

Log messages from individual process instances are always sent to the Integration Server server.log file. However, these messages are mixed together with Dynamic Business Orchestrator messages, as well as messages from other process instances, so it can be hard to find the specific messages you are looking for.

The Dynamic Business Orchestrator also supports process instance diagnostic logging for individual process models. When you enable process instance diagnostic logging for a process model, the process instance log messages are sent to both the server.log file and to a separate process instance log file. You can then access the process instance log file to see the messages from an individual process instance.

To enable or disable process model instance diagnostic logging

1. In My webMethods: **Navigate > Applications > Administration > Business > Business Processes**.
2. Navigate to the process model, for which you want to enable or disable diagnostic logging.
3. Edit the webMethods-executed model version or externally executed process model that you want to enable for analysis.

   Note: Enabling this option increases the processing overhead for all instances of this process model, which may have an impact on performance. You are advised to disable this option as soon as you have completed your diagnostic activities.

4. In the **Diagnostic Settings** window:
   - Select the **Diagnostic Logging Enabled** check box to enable process instance logging.
   - Clear the **Diagnostic Logging Enabled** check box to disable process instance logging.
5. Click **Save**.

Viewing a Process Instance Diagnostic Logging File

When a process model is enabled for process instance diagnostic logging, a log file is created in the directory *Software AG_directory*/IntegrationServer/serverName/instances/*instance_name* /packages/WmDBO/log, with a file name of *processInstanceID*.log. If the process is running on multiple servers (for example, in a distributed environment), a diagnostic log file is created on each server that runs the process. In this case, you must view all of the instance logs to get a complete picture of the instance activity. The log file name is the same on each server.

You can view the file in any text editor, or you can dynamically monitor the file in a command session using the `tail` command. The `tail` command is available on all

Linux and UNIX systems, and on some Windows systems. If your Windows system does not offer the `tail` command, you can download it from the following locations:

- As part of the "Windows Server 2003 Resource Kit Tools" available from Microsoft.
- As an executable from "Sourceforge".

Message entries are formatted as follows:

[*timestamp messageID threadID*] *processInstanceID:iteration stepID message*

All messages that are not pertinent to the process instance (for example, correlation of an incoming document) are sent to the server.log file.

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.

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

Column	Description
	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.
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 **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

Column	Description
	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**.

Viewing Stages in the Process Diagram

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.

To view a stage in the process diagram, select any row in the stage list as follows:

- Click the option button next to stage name to select a stage row. The icon becomes active and the milestone pins are rendered in diagram.

- To disable the pin display in the diagram for a stage row and keep the row selected, click the  icon.
- To disable the pin display in the diagram for a stage row and clear the row selection, click the option button  .
- For a selected stage row with pin display disabled, click  to display the pins again.
- Click the option button  of another stage row to switch stage selection. The milestone pins on diagram are updated for the newly selected stage

About Synchronizing Stages and Events with Software AG Designer

You can create, modify, and delete stages, and enable/disable EDA events, in two locations:

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

In both cases, any changes to the stage runtime settings in a process model 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 runtime settings are always synchronized between the two locations.

In Monitor, you must consider the following:

- When you open the Edit Process page to view the process model settings in the Process Stages and EDA Events window, the stage and event settings saved in the Process Audit database are retrieved and displayed.
- When you save the process model stage and event settings on Edit Process page, the settings are written to the Process Audit database, overwriting whatever stage and event settings are stored there.

Designer writes stage and event settings to the database when the process model is built and uploaded, overwriting whatever stage and event settings are stored there.

If a Designer user and a Monitor user modify the database stage or event settings at the same time in Designer and My webMethods respectively, the Designer user could overwrite the updates of the Monitor user and vice versa.

A **Synchronize** button is available in Designer for both stages and events, enabling the Designer user to apply the current database stage settings to the process model in Designer. You are advised to establish procedures to ensure that stage settings event settings are managed without conflict between Designer users and Edit Process page users.

To help ensure that you are working with the latest settings, you are advised to either refresh the Edit Process page or re-open the Edit Process page immediately before modifying and saving stage and event settings.

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**.

VI Monitoring Process Instances

Controlling Process Instances

Changing the Status of a Process Instance

You can change the status of Dynamic Business Orchestrator process instances in the webMethods Business Console web user interface. You can perform the following actions on a process instance depending on its current status:

- Cancel
- Suspend
- Resume
- Restart

To change the status of 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. Select an action to change the status of the process 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.

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	Actions for Status Change
Running (101)	The process instance is running.	Cancel, Suspend
Completed (102)	The process instance is completed.	None
Cancelled (103)	The process instance is cancelled. No further execution is permitted.	None
Terminated (104)	The process instance was terminated abnormally. The terminated status is a result of	None

Status	Description	Actions for Status Change
	a fatal runtime exception or an end terminate event. **Note:** A fatal runtime exception indicates problems with the process design.	
Suspended (105)	The process instance is suspended by a control action.	Cancel, Resume
Needs-Attention (106)	The instance is running, but one or more steps have unhandled errors that must be addressed before the instance can be completed.	Cancel, Restart
Interrupted (107)	The process instance was interrupted due to an interrupting boundary event or an interrupting event subprocess.	None

End Terminate Event Behavior

In Dynamic Business Orchestrator, you cannot configure end terminate events. End terminate events result in the abnormal termination of a process and set the process instance status to Failed. Any other running tracks of the instance are also terminated. The process instance cannot be restarted.

Process Step Statuses

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

Status	Description
Running (101)	The step is running.
Completed (102)	The step was completed normally.
Interrupted (107)	The step activity was interrupted by a boundary event with an interrupting property.
Failed (108)	The step returned an exception and did not complete normally. The status of the process instance that holds this step is determined by the type and exception handling in the

Status	Description
	process model. Dynamic process injection is available for a failed step.
Paused (109)	The step execution reached a breakpoint. The execution can be resumed in the Business Console user interface. For more information about resuming a paused step see " Dynamic Process Injection in Dynamic Business Orchestrator" on page 70.
Redirected (110)	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.
External Id (111)	An external ID was logged for the step. For a user task step, this is the task instance ID from Task Engine.

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** in order to make it clear that a goto operation was invoked on the paused step. The process instance status is changed to **Running** to indicate that no action is required 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**.

Setting Breakpoints to Pause Process Instance Execution

Before injecting a process, you must interrupt the normal process execution flow of a process instance by using one or multiple*breakpoints*. You set and remove breakpoints for a step at runtime in the Business Console user interface.

You can set a breakpoint on any step of a process instance with status Running or Needs-Attention. 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 a break point 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, locate the step or steps for which you want to add a breakpoint, and click the Set Breakpoint icon.

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**.

After a process instance pauses at a breakpoint, you can inject a process. When injecting a process, you select from the list of active process types and select one of the following options:

- **Contiunue** - after injecting the call activity, 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.

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 days are the same - all days of the week contribute to the same average.
- Work days and weekend days -weekdays contribute to one average, while weekend days contribute to a separate average.
- All days are different - 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 to Path Forecasting for a Process Instance

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 following **Analytical Engine Settings**:

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. The alerts and error notifications can

be moved to history, which clears all notifications and moves them to the notifications history.

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

You can

To move process instance alerts and error notifications to history:

1. Go to the **Processes** tab of webMethods Business Console.
2. Click the Alerts button.
3. Click **Move To History**.

Note: Moving notifications to history dismisses them from the Alerts window.

VII 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.

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.

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

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.

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.
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.

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

Output Parameters

None.

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.

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.

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* , 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.