opentext™

OpenText™ Output Transformation Server

User Guide

This document provides information about the features and functionality of OpenText Output Transformation Server.

VDTOTS240200-UGD-EN-2

OpenText™ Output Transformation Server
User Guide

VDTOTS240200-UGD-EN-2
Rev.: 2024-May-14

This documentation has been created for OpenText™ Output Transformation Server CE 24.2.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Introduction

1.1 Introduction

OpenText Output Transformation Server enables an organization to streamline their information supply chains by eliminating both the deficiencies and unnecessary links when sharing content between multiple sources. This process allows for the high speed integration, composition, translation, transformation, indexing, repurposing, archiving, retrieval, routing, printing, delivery, and presentment of structured and unstructured data throughout the enterprise and beyond. It makes possible the flexible sharing of previously inaccessible strategic information assets with customers, employees and also trading partners.

Output Transformation Server is installed with one or more of the following OpenText products: Output Transformation Designer, Data Transformation Engine, ebXML Messaging, and Output Transformation Engine. The tabbed pane layout within the Output Transformation Designer interface allows you to open multiple OpenText products simultaneously and switch from one resource to another.

The OpenText products share the following features:

- **Output Transformation Server** engine: Allows the high performance OpenText transformation products, events and process flows to be deployed to a JEE application server. Once deployed, the projects can be run either from the events themselves, or they can be run from a session bean, JMS queue, SOAP/Web service, or interactively from Output Transformation Designer. There is also a common API which allows projects to be called directly from Java.
- **Package and Deploy wizard**: Helps you build the JEE EAR file that is deployed to a supported application server.
- **Output Transformation Designer**: Contains a fully functional Java editor which enables the creation of plugins and processes; allows you to configure the products and deploy the project files and resources to the application server.
- **OpenText Output Transformation Server Manager**: Available for web-based control.

1.1.1 Target Users

With Output Transformation Server being an amalgamation of multiple OpenText technologies into a single framework with a unified interface, the varying needs of a wide range of users can be met. Systems Integrators, Business and Systems Analysts, Technical Consultants, Information Architects and Software Developers are only a few types of users. The commonality is their need to work with information supply chains, and Output Transformation Server is versatile enough to enable sharing between partners and customers that was, until now, difficult to implement and unrealizable within a single product.

Chapter 2

System Configuration

The system configuration file contains all of the information that relates to the way Output Transformation Designer and Output Transformation Server function.

2.1 System Configuration Parameters

The System Configuration file is used to view and edit the system configuration parameters. This group of parameters sets system-wide properties that affect all aspects of your Output Transformation Server projects. Some of these values can be overridden on a per-project basis.

When you first install the application, a default set of parameters, which is suitable for most projects, is loaded. It can be found in `<install_home>\initialFiles\common\_configs` and is named `default.xSystemConfig`. System-wide configuration modifications are made using this file.

The descriptions in this section explain the default settings for the parameters, as well as information about the parameters and how to best use the settings under normal conditions. Additionally, within Output Transformation Designer, when the System Configuration file is opened in a tab, basic descriptions of each parameter are displayed in the parameter descriptions window when the parameter is selected.

2.1.1 System ID

This parameter indicates the unique system identifier.

2.1.2 JobVariables

System job variables are defined with this parameter. Variables are listed in name-value pairs, and can be accessed like any other job variable. System variables are available to all jobs but will be overridden by job level variables (i.e. if a job variable of the same name is defined at the job level).

2.1.3 ThreadPool

This parameter specifies the thread pool used by all Output Transformation Server events and services.

2.1.3.1 MinThreads

Specifies the minimum number of threads to use, when needed.

2.1.3.2 MaxThreads

Specifies the maximum number of threads to use, when needed. Enter 0 for an unlimited number of threads.

2.1.3.3 KeepAliveTime

Indicates the amount of time, in milliseconds, before an idle thread will be eliminated and the garbage collected.

2.1.4 JobManager

The JobManager parameters control job handling. For more information, see "JobManager Parameters" on page 18.

2.1.4.1 JobManager Parameters

This section describes parameters that help control the overall way job submission is managed on your Output Transformation Server system. These parameters are found in the **JobManager** section of the system configuration file (`.xSystemConfig`). For more information, see "System Configuration" on page 17

2.1.4.1.1 MaxConcurrentJobs

The **MaxConcurrentJobs** parameter defines the maximum number of concurrent jobs that can be processed at one time. Typically, the most significant impact on performance is determined by the number of concurrent jobs. We suggest you test with N concurrent jobs, where N is the number of CPUs in your server. Most production machines will easily be able to handle more. For more information, see "Modifying the Output Transformation Server system configuration settings for higher performance" on page 53.

2.1.4.1.2 CheckQueuedJobs

The **CheckQueuedJobs** parameter defines options to control job submission based on the number of queued jobs. If enabled, new jobs will not be queued while the number of queued jobs is greater than or equal to the value of **MaxQueuedJobs**. Under heavy load, jobs will be intentionally failed if the queue is full. This option is disabled by default.

Benefit

If **CheckQueuedJobs** is not enabled (the default setting), Output Transformation Server will queue all submitted jobs. Memory is at a premium in large-scale implementations where printstreams are passed entirely in-memory (typically from a large ECM document or image archive), since each queued job requires memory for the input that is therefore not available for processing. If the system is being overloaded with job submissions, there will be a possibility of an OutOfMemoryError occurring, which could result in a system crash.

Limiting the number of queued jobs allows the job submitter application to quickly reclaim used memory if, based on the configuration settings, Output Transformation Server decides that it will not be able to handle the job given the current load.

Recommended Configuration

The maximum number of queued jobs selected depends on your situation and the trade-off that you are willing to make between system stability and the number of failed jobs that will not produce output. Taking this into account, set **MaxQueuedJobs** relative to the maximum number of concurrent jobs, defined by the **MaxConcurrentJobs** parameter. There are three configuration scenarios:

- **MaxQueuedJobs > MaxConcurrentJobs**

  The more MaxQueuedJobs exceeds MaxConcurrentJobs, there will be a greater chance of a job being available when a running job completes.

  A reasonable configuration in a Dynamic Retrieval scenario would be MaxQueuedJobs=200 and MaxConcurrentJobs=10. The application that is submitting the job to the engine would have to include logic to deal with rejected jobs in the event of a spike in load.

- **MaxQueuedJobs = MaxConcurrentJobs**

  There should always be a job available when a job completes. This is likely to be a better configuration for an environment with larger jobs where memory is a concern. However, in a high load and small job size environment, a number of jobs would not get a chance to be processed.

- **MaxQueuedJobs < MaxConcurrentJobs**

  In this situation, there may never be a job available when a running job completes. This would be a less than optimal configuration.

Notes

In realistic situations where jobs are submitted on average over a longer period of time, this parameter will work as designed. For performance reasons, Output Transformation Server job submission methods are only loosely synchronized. Due to this design, a large number of jobs being submitted concurrently may not be accurately controlled by the **MaxQueuedJobs** parameter.

If the user chooses to synchronize their own job submission code, and the submitter waits for the job to complete, multiple jobs will not be submitted at once and will instead be processed sequentially, negatively impacting performance, whereas if the submitter chooses not to wait for the job to complete, this leaves the handling of the output stream entirely with Output Transformation Server.

2.1.4.1.3 CheckHeapUsage

CheckHeapUsage defines options to control job submission based on Java memory heap usage. If enabled, **MaxHeapPercent** defines the maximum percentage of the heap to use before failing new jobs (percentage of HeapSize or value specified by -Xmx). New jobs will not be queued while the percentage of the heap used is greater than or equal to this value. This helps prevent jobs that will not run soon from using all the available memory.

The **HeapSize** parameter indicates the maximum Java heap size in megabytes (as specified by the -Xmx option).

Benefit

By default (i.e. if **CheckHeapUsage** is not enabled), Output Transformation Server uses as much of the heap as it requires. Throttling the amount of memory that Output Transformation Server can use (out of the total available) helps to prevent the JVM from getting into situations where the maximum heap size is reached and a subsequent garbage collection is unable to reclaim enough memory to continue, resulting in OutOfMemory errors. This prevents a single job from using all the available memory.

Recommended Configuration

The configuration chosen depends on your situation and the trade-off that is willing to be made between system stability and the number of failed jobs that will not produce output.

Notes

For performance reasons, heap usage is checked only upon job submission.

2.1.4.1.4 JobProfiling

The **JobProfiling** parameter can be used to help profile job heap usage and set an optimal configuration for your environment. If enabled, a number of jobs are run (set by the **JobCount** attribute) and heap usage statistics are recorded at a given interval (set by the **SleepInterval** attribute). The resulting statistics are factored into a calculation that determines and sets the number of jobs Output Transformation Server thinks can be handled concurrently (**MaxConcurrentJobs**):

Max Heap Size * Max Heap Percent - Post-profiling Engine Size = Amount Available

Amount Available / Maximum Job Size = Maximum Concurrent Jobs

This calculation takes into consideration the maximum heap percent and size set by the **CheckHeapUsage** parameter, if enabled (for example, a lower value for **MaxHeapPercent** means a lower value for **MaxConcurrentJobs**).

A warning is logged to the system report when **MaxConcurrentJobs** has been overridden.

Benefit

This parameter provides a very useful diagnostic tool in determining the load the engine can handle in terms of concurrent jobs. Furthermore, it will override the existing setting to ensure that an optimal value is used.

2.1.5 ServiceAutoStart

This parameter allows you to specify which services or events will start automatically during system initialization. For more information, see "ServiceAutoStart Parameter" on page 21.

2.1.5.1 ServiceAutoStart Parameter

The **ServiceAutoStart** parameter is found in the system configuration file, and allows you to specify which services or events will start automatically during system initialization.

To configure the **ServiceAutoStart** parameter:

1. Open the system configuration file and locate **ServiceAutoStart** in the list of parameters.

   The **Value** column indicates whether any services have already been set to start automatically.

   For more information on the system configuration file, see "System Configuration" on page 17.

2. Click the **Ellipsis**,  , to the right of the **Value** column.

   The **Property - ServiceAutoStart** dialog is displayed.

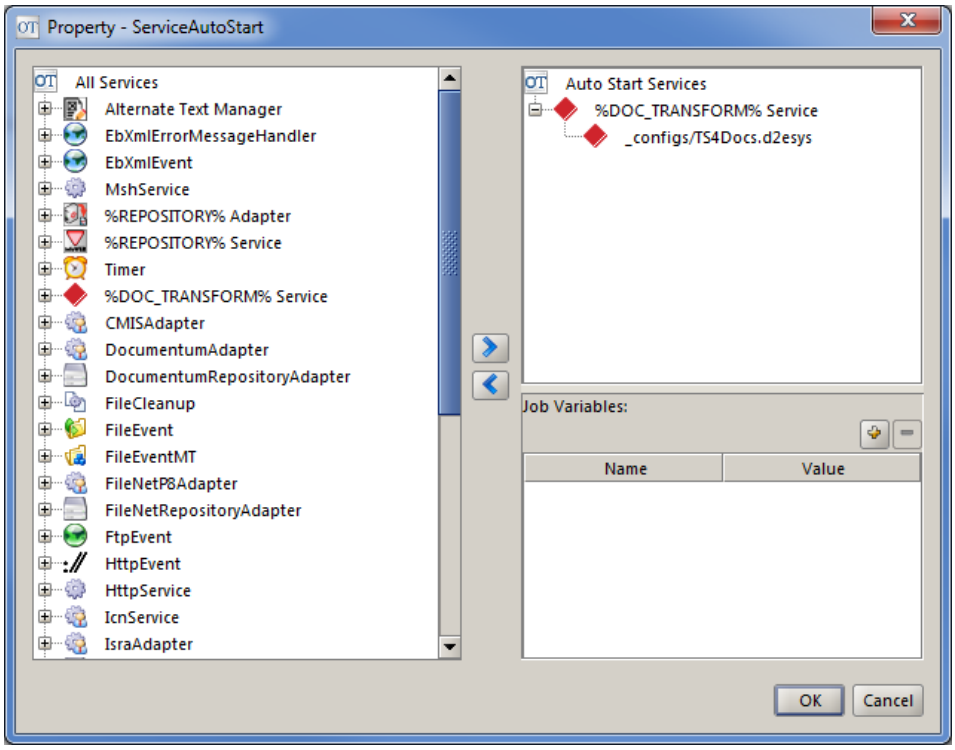


Figure 2-1: Property - ServiceAutoStart dialog

The **Property - ServiceAutoStart** window contains the following panes:

- **All Services.** Lists all the services and events available in your file system. Expand each category to display the individual services and events.
- **Auto Start Services.** Displays the services selected to start automatically. You can add to this list by highlighting an item in the **All Services** pane and clicking the button. Similarly, you can remove an item from the list by clicking the button.
- **Job Variables.** Allows you to define job variables for a service or event. You can add a job variable by selecting the service or event in the **Auto Start Services** pane, clicking the button, and entering a **Name** and **Value** for the variable. Similarly, click the button to remove the job variable.

Note: Job variables entered here will also apply when running the event manually. For more information, see "Running an Event" on page 162.

3. Click **OK** to save your changes.

2.1.6 LogProfile

You can configure your system logging profile with this parameter. For more information, see "Third Party Connectors – Microsoft" on page 156.

2.1.6.1 Overwrite

Specifies whether a log profile can overwrite the targets defined in the previous log profile.

2.1.6.2 LogProfileRef

Allows you to select from available log profiles to define the system logging configuration. Click on the **Ellipsis** button to add or remove log profiles.

2.1.7 ComponentPoolManager

The ComponentPoolManager parameters allow you to define job handling within the context of specific runnable components. For more information, see "ComponentPoolManager Parameters" on page 23.

2.1.7.1 ComponentPoolManager Parameters

The **ComponentPoolManager** allows you to define how a job should be handled within the context of a specific runnable component. After setting up the ComponentPoolManager, each component defined in your project can be pooled by the engine. This improves performance by streamlining the load process to include only the necessary components for your project.

2.1.7.1.1 PoolEntry

This parameter specifies a pool of instances for a particular Output Transformation Server runnable, which can be a runnable component, a transformation, or a process flow.

2.1.7.1.2 ComponentName

This setting indicates the runnable component's configuration name and path relative to the Output Transformation Server file system root.

For example: `_sample/OutputTransformation/processFlow/Isv2File.xProcessFlow`

2.1.7.1.3 AliasName

This parameter specifies the alias (short) name which can be used to call the runnable component.

2.1.7.1.4 UserDescription

This parameter provides a description of the project.

2.1.7.1.5 InitialPoolCount

This value indicates the number of instances that will be created at startup. For values greater than 0, the component pool will be created and populated with the specified number of component instances. If the value is set to 0, then the component pool will be created empty. The default value is 1.

2.1.7.1.6 MaxPoolCount

This parameter specifies the maximum number of instances that can be created. Set to 0 for unlimited. If the number of requests exceeds this value, the job ticket will be rejected. If not set to 0, this value must be greater than or equal to the **InitialPoolCount** value.

2.1.7.1.7 MinEvictableIdleTimeMillis

This parameter specifies the minimum amount of time, in milliseconds, that an object may sit idle in the pool before it becomes eligible for deletion. The default value is 60000 milliseconds.

2.1.7.1.8 TimeBetweenEvictionRunsMillis

This parameter specifies the amount of time between eviction runs. The default value is 60000 milliseconds.

2.1.7.1.9 NumTestsPerEvictionRun

This parameter indicates the number of objects to examine per run in the idle object evictor. The default value is 3.

2.1.8 JobAuditService

The JobAuditService allows you to audit jobs once they are executed. For more information, see "JobAuditService Parameters" on page 25.

2.1.8.1 JobAuditService Parameters

The JobAuditService allows you to audit or query jobs once they are executed. If enabled, this service provides statistics about executed jobs, including start time, running time, the name of the executed process flow, the name of the event calling the process, the output size, and whether there were any errors.

2.1.8.1.1 Enabled

Select this parameter to enable job auditing services for executed jobs. Note that enabling JobAuditService may impair the performance on your machine. For more information, see "Modifying the Output Transformation Server system configuration settings for higher performance" on page 53.

2.1.8.1.2 AuditTarget

This parameter defines the destination of the job audit records. The possible options are:

- **LOG**. Records are written into log files located on the file system. All files will be written to either:

  `<install_home>\initialFiles\common\_stats\<Day_yymmdd.stats>\<Hour_xx>.stats`, or

  `<install_home>\initialFiles\common\_stats\` (if running standalone Output Transformation Server from Output Transformation Designer)

- **DATABASE**. Records are written into a database. If DATABASE is selected, the **PersistentStorageName** must be specified.

2.1.8.1.3 PersistentStorageName

This parameter specifies the name for the persistent storage. This should be the same value that is specified for **PersistentStorageName** in the PersistentStoragePool parameter. If this value is left blank, the database audit service will be disabled and the log-based service will be enabled. This value is only required if **AuditTarget** is set to DATABASE.

Note: If you have multiple persistent storage connections defined in the PersistentStoragePool, only one can be used with the JobAuditService at any given time.

2.1.8.1.4 History

Select this parameter to collect the job status history for sub-process flows. If this parameter is not selected, then only the job status history for the main process flow will be recorded. This applies to both LOG and DATABASE type audit targets.

2.1.8.1.5 CreateHumanReadable

This parameter controls whether or not job status information will be written into a human-readable log file (i.e. a file with a more readable format), in addition to the Job Stats CSV file. If this parameter is enabled and **AuditTarget** is set to LOG, then the job status will also be written to the Output Transformation Server debug log.

2.1.8.1.6 DaysToLive

This parameter specifies the number of days to keep the audit record files. JobAudit file cleanup typically occurs at midnight on the last day specified. Set to 0 (zero) to disable the deletion process.

2.1.9 SystemSchedule

The SystemSchedule parameters control the schedules for jobs and services. For more information, see "SystemSchedule Parameters" on page 26.

2.1.9.1 SystemSchedule Parameters

The **SystemSchedule** lists the jobs and services scheduled for execution by the system. These schedules can be defined with the SystemSchedule parameters, or with the Scheduler feature of the deployed server. Schedules can be added, deleted, or edited using either interface. For more information, see "Scheduler" on page 285

2.1.9.1.1 JobSchedule

This parameter displays the specified schedule for a list of jobs.

2.1.9.1.2 Job

These settings provide the schedule details for a specific job with the following properties:

- **Name**. Specifies a unique name for the scheduled job.
- **Description**. Describes the purpose of this scheduled job.
- **ConfigFile**. Specifies the alias or component name for the job to be executed.
- **Enable**. Indicates whether the scheduled job is turned on or off. By default, the job is enabled.
- **JobVariables**. Defines the names and values of any job variables required by the scheduled job in order to run properly.

- **Schedule**. Specifies when to start the job. By default, the job starts each day at 00.00.00 o'clock (or midnight). Configure the following settings to define the start date and time for the scheduled job:
  - **Seconds/Minutes/Hours**. Collectively define the start time. The range of valid values is 0-59 for Seconds and Minutes, and 0-23 for Hours. The default value for each setting is 0.
  - **DaysOfMonth**. Specifies the days of the month that the job should run. This field accepts values in the range of 1-31. The default value is every day of the month.
  - **Month**. Indicates which month(s) to run the job. You can enter a number from 1 to 12. The default value is every month of the year.
  - **DaysOfWeek**. Specifies the days of the week that the job should run. Enter a number from 1 to 7, with 1 = Sunday, 2 = Monday, etc. The default value is every day of the week.
  - **Year**. Indicates which year(s) to run the job. The range of valid values is 0-9999. The default value is every year.

Note: The string may include wildcards. For examples of configured schedules, see "Scheduler" on page 285.

2.1.9.1.3 ServiceSchedule

This parameter displays the specified schedule for a list of services.

2.1.9.1.4 Service

These settings provide the schedule details for a specific service with the following properties:

- **Name**. Specifies a unique name for the scheduled service.
- **Description**. Describes the purpose of this scheduled service.
- **ConfigFile**. Specifies the alias or component name for the service to be executed.
- **Enable**. Indicates whether the scheduled service is turned on or off. By default, the service is enabled.
- **JobVariables**. Defines the names and values of any job variables required by the scheduled service in order to run properly.
- **StartSchedule**. Specifies when to start the service. By default, the service starts each day at 00.00.00 o'clock (or midnight). Configure the following settings to define the start date and time for the scheduled service:
  - **Seconds/Minutes/Hours**. Collectively define the start time. The range of valid values is 0-59 for Seconds and Minutes, and 0-23 for Hours. The default value for each setting is 0.

  - **DaysOfMonth**. Specifies the days of the month that the service should run. This field accepts values in the range of 1-31. The default value is every day of the month.
  - **Month**. Indicates which month(s) to run the service. You can enter a number from 1 to 12. The default value is every month of the year.
  - **DaysOfWeek**. Specifies the days of the week that the service should run. Enter a number from 1 to 7, with 1 = Sunday, 2 = Monday, etc. The default value is every day of the week.
  - **Year**. Indicates which year(s) to run the service. The range of valid values is 0-9999. The default value is every year.

- **StopSchedule**. Specifies when to stop the service. By default, the service stops each day at 23:59:59 o'clock. Configure the following settings to define the start date and time for the scheduled service:

  - **Seconds/Minutes/Hours**. Collectively define the stop time. The range of valid values is 0-59 for Seconds and Minutes, and 0-23 for Hours. The default value for each setting is 0.
  - **DaysOfMonth**. Specifies the days of the month that the service should stop. This field accepts values in the range of 1-31. The default value is every day of the month.
  - **Month**. Indicates which month(s) to stop the service. You can enter a number from 1 to 12. The default value is every month of the year.
  - **DaysOfWeek**. Specifies the days of the week that the service should stop. Enter a number from 1 to 7, with 1 = Sunday, 2 = Monday, etc. The default value is every day of the week.
  - **Year**. Indicates which year(s) to stop the service. The range of valid values is 0-9999. The default value is every year.

Note: The string may include wildcards. For examples of configured schedules, see "Scheduler" on page 285.

2.1.10 PersistentStoragePool

The PersistentStoragePool parameters help control the overall way Output Transformation Server handles persistent storage for your projects. For more information, see "PersistentStoragePool Parameters" on page 29.

2.1.10.1 Creating a New Persistent Storage Pool Connection

To create a new persistent storage pool connection:

1. On the File Systems tab in Output Transformation Designer, double-click your active **Output Transformation Server system configuration file** (`default.xSystemConfig`, unless you are running your own custom system configuration file).

   The system configuration file opens on a new tab in the **Development** window.

2. Expand the **PersistentStoragePool** section.

3. Right-click **PersistentStorage** list parameter () and from the context menu that appears, select **Add**.

   A new **PersistentStoragePool** instance is added with a randomly generated name. (You can change the name using the **PersistentStorageName** parameter.)

4. Expand the PersistentStoragePool instance to see its parameters, and select the **Enabled** check box to turn on this instance of the persistent storage pool.

5. Configure the remaining persistent storage pool parameters for your database. (See below for more information on the parameters.)

6. **Save** the system configuration file and your persistent storage instance is ready to use.

   The **Restart the system?** dialog appears.

7. Since you modified the active system configuration, the engine must be restarted in order for your new settings to take effect. Click **Yes** to restart the system.

   The **Building Environment** dialog displays and the system configuration with your new settings is loaded. Once this message disappears, the system is ready for use.

2.1.10.2 PersistentStoragePool Parameters

This section describes persistent storage for your projects in Output Transformation Server. If persistent storage is enabled, a database connection is maintained so that the open connection can be used again in the future whenever communication with the database is needed.

2.1.10.2.1 PersistentStorageName

This parameter specifies the name of the persistent storage instance.

2.1.10.2.2 Enabled

If you want to use the persistent storage pool for your project, you must enable it with this parameter. By default, persistent storage is **disabled**.

2.1.10.2.3 NumberOfTries

The application can automatically retry getting a valid connection if the initial attempt is unsuccessful. The number of times to retry is indicated by this parameter while the length of the delay, in milliseconds, between each connection attempt is specified by the **TimeBetweenRetries** parameter. The default number of times to retry is **1**.

2.1.10.2.4 TimeBetweenRetries

This parameter specifies the length of the delay, in milliseconds, between each connection retry after a failed initial attempt. The default is **500**. The number of retries is determined by the **NumberOfTries** parameter.

2.1.10.2.5 TestCheckOutConnection

To have the validity of your connections tested by the system periodically, you can enable this setting. If enabled, a test is performed at every connection checkout to confirm that the connection is valid. By default this test is **disabled**.

2.1.10.2.6 Connection

The Connection parameter contains several settings related to the various database connection types. The following parameters are available:

- **Type**. The Type parameter specifies the type of connection that you are using to connect to the database. You can choose from either **JDBC** or **JNDI** with the default set to **JDBC**.
- **JdbcConnection**. This set of parameters only needs to be configured if your database connection is via JDBC. Prior to configuring these settings, you must create the database and add the proper JDBC driver to the classpath.
  - **Url**. Type in the URL for the JDBC server as the value for this parameter. You also must include any extra connection type information with the URL. For example, `jdbc:sqlserver://<host>:<port>`; `databasename=<DB_Name>`; and `SelectMethod=cursor` are all acceptable formats of URL paths.
  - **JdbcDriver**. Choose the Java driver for the database by clicking the **Ellipsis**. The driver must be accessible from your production machine.

  - **Debug**. Contains options for logging JDBC driver activity for debugging. All of these debug log messages are written to the console only and do not appear in the `all-debug.textreport` or `application.log` files like other log messages.
    - **Enabled**. Specifies whether debug logging is active.
    - **Select**. Denotes the logging of Select SQL statement messages.
    - **Insert**. Denotes the logging of Insert SQL statement messages.
    - **Delete**. Denotes the logging of Delete SQL statement messages.
    - **Update**. Denotes the logging of Update SQL statement messages.
    - **ClassFilter**. Designates the logging of SQL statements originating from the specified class filter. This value can be left empty.

- **JndiConnection**. This set of parameters only needs to be configured if your database connection is via JNDI. Prior to configuring these settings, you must remember to create the database, set the data source from your WebSphere Liberty application server, and if required for your particular application server type, add the necessary JAR files to the classpath.
  - **Url**. Type in the URL for the JNDI provider. This parameter is optional. You must adhere to the correct format for your server type:
    - **WebSphere**. `url=iiop:<address>`
  - **DataSource**. DataSource specifies the corresponding JNDI name for the data source.
  - **ClassName**. Type in the class name for the JNDI InitialContextFactory as the value for this parameter, but depending on your type of application server you may not need to enter a value. You can locate the class name for your InitialContextFactory by looking it up in the `<install_home>\sampleApplication\OutputTransformation\bin\server.info.*` file that corresponds to your application server type. For example, the default InitialContextFactory for WebSphere is `java.naming.factory.initial=com.ibm.websphere.namingWsnInitialContextFactory`.
- **UserName**. The value for this parameter specifies the user name required to log in to the database.
- **Password**. The value for this parameter specifies the password for the corresponding user name to log in to the database.
- **ConnectionPool**. This set of parameters only needs to be configured if you are using C3P0 connection pools.
  - **MinSize**. Indicates the minimum number of connections to be held in the pool. The default size is **5**.
  - **MaxSize**. Indicates the maximum number of connections to be held in the pool. The default size is **20**.

- **Timeout**. Pooled connections can be discarded after a certain period of inactivity. If you want your connections to expire, specify the amount of time, in seconds, before an unused connection is discarded. Setting this parameter to **0** disables the tests and means that idle pooled connections will never expire.
- **MaxStatements**. Since creating prepared SQL statements is a time consuming operation, the PreparedStatement cache is used to store these SQL statements for reuse at a later time. This parameter designates the size of C3P0's PreparedStatement cache; simply enter the number of SQL statements you want the cache to hold. Setting this parameter to **0** disables statement caching.
- **IdleTestPeriod**. IdleTestPeriod stipulates the length of time, in seconds, between tests by the C3P0 provider on all idle and pooled but unchecked out connections. If the test is failed, the connection is removed from the pool. Setting this parameter to **0** will disable the tests.
- **AcquireIncrement**. If you want C3P0 to try and acquire simultaneous connections when the pool is exhausted, enter the number of simultaneous connections as the value. The default is **1**.
- **AcquireRetryAttempts**. This field indicates the number of times the connection pool will retry making a connection if the initial attempt is unsuccessful. The default is **30**. (The length of the delay, in milliseconds, between each connection attempt is specified by the **AcquireRetryDelay** parameter.)
- **AcquireRetryDelay**. This field specifies the length of the delay, in milliseconds, between each connection retry after a failed initial attempt. The default is **1000**. (The number of retries is determined by the **AcquireRetryAttempts** parameter.)

2.1.10.2.7 DatabaseType

The SqlDirect parameter is where you designate the particular SQL dialect used by a Persistent Storage Manager. You can choose from **HypersonicSQL**, **MsSQL**, **Oracle**, or **PostgreSQL**.

2.1.10.2.8 HibernateProperties

This set of parameters contains additional configuration settings for the Hibernate feature. Any settings used here will override the default values.

The **Name** parameter indicates the name of the Hibernate property, while the **Value** parameter is where you enter your Hibernate property values.

2.1.10.2.9 MappingResources

If you are using the Hibernate feature, use this parameter to specify the Hibernate XML mapping documents. These mapping files must also be in the classpath.

If your project contains the Alternate Text Manager component and you are using the PersistentStorageConnectionParm database type for the ConnectionType parameter, there are several specific hibernate mapping XML files you must add as mapping resources in order for the Alternate Text Manager tables to be created in your database. These XML files exist within JAR files that are installed with the product. You must add the following files to the MappingResources parameter:

- `com/xenos/alttext/db/Image.hbm.xml`
- `com/xenos/alttext/db/Alternatetext.hbm.xml`
- `com/xenos/alttext/db/Tag.hbm.xml`
- `com/xenos/alttext/db/Imagetag.hbm.xml`
- `com/xenos/alttext/db/ImageLock.hbm.xml`
- `com/xenos/alttext/db/Locale.hbm.xml`

For more information about the Alternate Text Manager component , see "Alternate Text Manager" on page 250.

2.1.11 ClusterManager

The ClusterManager parameters allow you to configure server cluster properties. For more information, see "Setting Up Server Clustering and Load Balancing" on page 40.

2.1.12 AuthenticationEngine

The AuthenticationEngine allows you to implement a selected authentication scheme and control the user authentication process.

2.1.12.1 Using User Credential Files

You can create your own user credentials file to identify users who want to gain access to your servers.

Creating a User Credentials File

The user credentials file contains accreditation information that is used to endorse the identity of users who try to gain admission to the Output Transformation Server Authentication Layer. Although you can opt to run our products with no user authentication whatsoever, it is highly recommended that you implement this layer of security or any of the other options within the authentication layer to reduce the risk of unauthorized access to the server.

The system administrator must manually create the file themselves. A sample user credentials file is provided with your installation and can be used as a template for

your own user credentials file. If you want to use the sample as a basis for your own user credentials file, you can delete the default user accounts and groups that are already in the file. The sample file can be found at:

```
<install_home>\initialFiles\common\_resources\authentication\user.
xAuthScheme
```

This file-based method of authentication is set up by default. It is enabled in the system configuration when the **AuthenticationEngine** > **AuthSchemeConfig** parameter is set to the file path location of the `user.xAuthScheme` file and the authentication scheme's **SchemeType** parameter is set to **AuthSchemeFileParm**.

A basic user credentials registry defines the user name and their password, which is automatically encrypted. Optionally, you can also add some metadata for each user. Before you begin to set up the user credentials file, you should determine the users, their passwords, and the groups to assign them to.

In terms of assigning users to groups, a user group is simply a collection of users in your environment, which can be useful since the collection of users is treated as one entity. For example, if you want to assign a particular role to a large number of users, instead of assigning the role to the users individually, you could assign the role to a group that contains all the users you want to have the same role. Group names are specified in the **UserGroups** section. There are several default groups, but you can add your own custom group by right-clicking the **UserGroups** parameter and from the context menu that appears, selecting **Add**.

Note: The user credentials file can be modified at any time even after being deployed to the server.

To create a basic user credentials file:

1. In Output Transformation Designer, open the **user.xAuthScheme** file for editing.

   The Authentication Scheme tab opens in the Development window.

2. In the right pane, ensure the **SchemeType** parameter is set to **AuthSchemeFileParm**, which indicates that file-based user credentials file is used.

3. Below the SchemeType parameter, the **Users** parameter stores a list of all existing user accounts and their properties. To add a new user, right-click the **Users** parameter and from the context menu that appears, select **Add**.

   A blank user properties entry is added to the bottom of the users list.

4. Expand the new entry and complete the following to set up a new user:

   - **Name**. Designates the user name for the user.
   - **Password**. Designates the password for the user. When typing in a password, it is hidden from view and is automatically encrypted upon saving the scheme.

5. Optionally, within the new user's node is the **Properties** section you can enter some basic details about the user. The default properties that can be added are shown in the **MappedUserProperties** parameter. When adding a new property to a user, you must use the value in the MappedUserProperties parameter as the Name for the property. The default values to call each mapped user property is shown, however, you can modify the values in the authentication scheme. The following user properties are available by default:

   - **FirstName**. Specifies the user's first name. This property can be called by the user properties using `givenName`.
   - **LastName**. Specifies the user's last name. This property can be called by the user properties using `sn`.
   - **Email**. Specifies the email address for the user. This property can be called by the user properties using `mail`.

   For example, to add a user's first name, under the user's Properties you must enter **givenName** as the **Name** and then type the user's name as the **Value**.

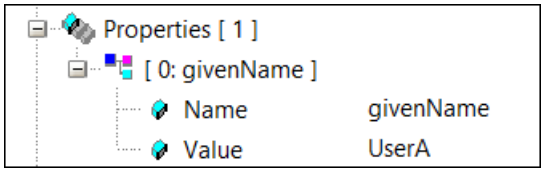


Figure 2-2: Sample file authentication scheme user properties with a first name entry

Note: By default, you can also add Mobile, Department, or Memo details to the user properties, but you must first assign a call value to them under MappedUserProperties.

6. Next, you must assign the user to a group. You can add a user to a group by assigning a value to the **MemberOf** parameter within a user's properties. This value must match one of the groups shown in the **UserGroups** section.

7. Repeat steps 3-6 until you have added properties for all the users.

8. Once you are finished configuring your users, save your changes.

   The user credential changes take effect immediately.

2.1.13 ComponentLookup

The ComponentLookup parameters define the list of Output Transformation Server ODA (Open Data Access) components. For more information, see "ComponentLookup Parameters" on page 36.

2.1.13.1 ComponentLookup Parameters

The **ComponentLookup** section of the System Configuration file allows you to configure Output Transformation Server IRunnable components to be compatible with the OpenText ODA (Open Data Access).

2.1.13.1.1 Groups

This parameter specifies a list of component groups.

2.1.13.1.2 GroupName

This parameter indicates the unique name for the component group. The GroupName **ODA** is reserved for ODA data sources, and only entries belonging to this group will be retrieved by the ODA.

2.1.13.1.3 ComponentEntries

Use this parameter to configure the following component entry parameters:

- **Alias**. This parameter specifies the unique alias name that identifies this component entry within the enclosing group.
- **Description**. This parameter provides a description of the component entry.
- **ComponentName**. This setting indicates the IRunnable component's configuration name and path relative to the Output Transformation Server file system root.
- **Input**. Use this parameter to define the input XDoc name.
  - **XDocName**. Specifies the name of the XDoc in the map.
  - **MapToUse**. Determines whether the input is taken from the input map or the result map. The default value is **inputMap**.
- **Result**. Use this parameter to define the result XDoc name.
  - **XDocName**. Specifies the name of the XDoc in the map.
  - **MapToUse**. Determines whether the result is taken from the input map or the result map. The default value is **inputMap**.
- **JobVars**. This parameter defines a list of the job variables that can be used by this component. Variables are listed in name-value pairs.

2.1.14 XDocProperties

The XDocProperties parameters define the system-wide XDoc properties. The size of each buffer and threshold in the XDoc properties can be specified in one of the following units: bytes (B), kilobytes (KB), or megabytes (MB).

2.1.14.1 InputBuffer

Specifies the size of the input buffer to use when reading in data.

2.1.14.2 ResultBuffer

Specifies the size of the result buffer to use when writing out data.

2.1.14.3 SpillThreshold

Determines the amount of data to store in memory before writing to a temporary file.

2.1.14.4 CompressThreshold

Determines the amount of data to accumulate before compressing it. The size must be less than or equal to the **SpillThreshold** value.

2.1.15 DebugConfig

The DebugConfig parameter indicates the debug file's configuration name and path relative to the Output Transformation Server file system root. The debug file defines debug behavior.

2.1.16 TransactionLog

The TransactionLog parameters control how Output Transformation Engine usage statistics are logged. For more information, see "TransactionLog Parameters" on page 37.

2.1.16.1 TransactionLog Parameters

The TransactionLog parameters control and activate the DocPages Summary log and CS Transaction log. These logs record Output Transformation Engine usage statistics. For more information, see "DocPages Summary Log" on page 38 and "CS Transaction Log" on page 39

2.1.16.1.1 IsActive

Select this check box to activate transaction logging. This option is enabled by default.

2.1.16.1.2 Root

Designates the root folder where the DocPages Summary log and CS Transaction log will be written. The default location is `${xenos.output}/logs`.

2.1.16.1.3 ActivateDocPagesSummary

Select this check box to activate the DocPages Summary log . This option is enabled by default.

2.1.16.1.4 TransactionLog Parameter Definitions

These are the TransactionLog parameters:

For more information on the DocPages Summary log and CS Transaction log, see "DocPages Summary Log" on page 38 and "CS Transaction Log" on page 39

2.1.16.1.5 DocPages Summary Log

The DocPages Summary log provides hourly tracking of OpenText Embedded Output Transformation Engine job activity by recording usage statistics related to the number of documents and pages generated. The log is written to the location defined by the TransactionLog > Root parameter.

Note: In order to use the DocPages Summary log, you must select the **IsActive** and **ActivateDocPagesSummary** check boxes under the TransactionLog parameter.

The format of the DocPages Summary log name is `DocPages-Summary-YYYY_instanceName.log`, where `YYYY` is the current year, and `instanceName` is the **SystemId** setting from the System Configuration file. This log is appended to each time the system starts. If the system is running at year's end, a new log will automatically start for the new year.

There are three record types in the DocPages Summary log:

The K (Key) Record

Example: K,DocPages,DocsParsed,PagesParsed,DocsGen,PagesGen,JobCount

This record always appears first and is written only once. It indicates the type of usage summary and the column headings which correspond to the data displayed in the H record.

The H (Hourly) Record

Example: S,20140219-15:15,DocPages

This record indicates the date and time when the log started and the usage summary type. In the example above, the log started on February 19, 2014 at

3:15 pm, implementing the DocPages usage summary, which tracks document transformation transactions.

The S (Start) Record

Example: H,hpnegadk,20140219-15,2,30,2,30,2

This record displays an hourly usage summary for every hour that the system is running:

- The first 8-character segment is a checksum value used internally to validate the usage shown.
- The subsequent 10 numbers indicate the day and hour being summarized. The example above was recorded on February 19, 2014 in the 15th hour (3:00 pm).
- The last 5 numbers show the total usage for the hour. Referring to the sample K record for the corresponding data headings, the example above shows that were 2 documents parsed, 30 pages parsed, 2 documents generated, 30 pages generated, and 2 jobs completed within that hour.

The H record is rewritten every time a job is run, as long as that job begins at least one minute after the previous job.

Related Links

- "Root" on page 38
- "TransactionLog Parameters" on page 37
- "System Configuration Parameters" on page 17

2.1.16.1.6 CS Transaction Log

The CS Transaction log records specific transactions that occur with the normal usage of OpenText Embedded Output Transformation Engine in order to track job activity. This log is used in conjunction with the DocPages Summary log to collect usage statistics.

The CS Transaction log is recorded in CSV format and is written to `<root>/transactions/YYYYMM/CS_Transactions_YYYYMMDD_instanceName.log` where `YYYYMM` and `YYYYMMDD` are the current year, month, and day, and `instanceName` is the **SystemId** setting from the System Configuration file. A new log is created at the beginning of each day, and new folders are created at the beginning of each month.

For more information, see "DocPages Summary Log" on page 38 and "System Configuration Parameters" on page 17.

2.2 Server Clustering

This section discusses various aspects of server clustering:

2.2.1 Setting Up Server Clustering and Load Balancing

The following sections describe server clustering and load balancing.

2.2.1.1 About Server Clustering

Server clustering enables you deploy multiple OpenText Output Transformation Server instances to multiple application servers and have them work together. This allows you to process information simultaneously on multiple servers, resulting in increased performance, availability and scalability across the application. Clustering enables the use of load balancing and failover capabilities, ensuring that if a server becomes overloaded or unavailable, there is little to no impact on the overall system.

A **webservice** cluster is defined by the SystemId and implemented by Output Transformation Server. A webservice cluster is used to run Output Transformation Server events and services. For webservice, load balancing/failover will be among cluster nodes that have the same SystemId. Clustering through webservice requires Cluster Manager settings to be configured at either the system configuration or component configuration level.

2.2.1.2 About Load Balancing

Load balancing is the even distribution of work amongst multiple nodes within a server cluster, optimizing efficiency and resource use. This technique also increases reliability through redundancy and results in greater scalability of services, since system performance improves with the addition of nodes to the cluster. Load balancing is part of the Cluster Manager setup described in the section below.

You can enable or disable load balancing for each event using the **LoadBalancing parameter**. See the event properties listed within each event type for more information.

2.2.1.3 Setting up Server Clustering

Server clustering can be configured in an environment with **WebSphere Liberty**.

2.2.1.3.1 Configuring WebSphere Liberty Server Clustering

1. Install and configure your WebSphere Liberty application servers. Your IBM WebSphere Liberty application server documentation contains detailed instructions on how to configure a multi-node environment. Their help provides configuration information, including notes related to configuring the application servers for a Output Transformation Server clustered environment.

2. Deploy Output Transformation Server onto one cluster. It will automatically be shared amongst the cluster.

2.2.1.3.2 Enabling Server Clustering and Load Balancing

1. Assign the **SystemID** for the cluster. Each node must be configured with the same properties in the Output Transformation Server system configuration file (`default.xSystemConfig`). Ensure SystemId is the same for each, as it defines the cluster ID.

2. Configure the cluster properties using the **Cluster Manager**, which is located in the system configuration file (default.xSystemConfig).

Enabled	Indicates whether to enable the cluster service. **Note:** Enabling the Configuration Manager from the Package and Deploy Wizard will automatically enable the cluster.
TreeCacheConfig	Specifies the configuration file used to manipulate the cluster manager. By default this is the `cluster-service.xml` file, located in the same folder that contains the default system configuration file: `<install_home>\initialfiles\common\_configs`. This XML file is used for configuring the server cluster. It is recommended to use the default settings. Since cluster implementation is based on the use of JGroups API, a thorough understanding of JGroups API is necessary before attempting to modify this file. Detailed documentation on JGroups API can be found at: `http://jgroups.org/ug.html`
ResponseTimeOut	Indicates the number of milliseconds that Coordinator waits for a response from a member of the cluster.
LoadBalancer	

LoadBalancerType	Indicates the type of cluster load balancing to be used for this configuration. Currently three types are supported: • The **Round-Robin** type supports an algorithm which cycles through a list of cluster nodes in order. For example, if there are two nodes in the cluster, the load balancer directs the first job to the first node in the list, the second job to the second node then starts over with the first node in the list. • The **Dynamic-Workload** supports an algorithm that provides a way of balancing workload by specifying that workload distribution should favor the node with the minimal number of queued and running jobs. For example, if there are two nodes with queue sizes of 10 and 5, running 4 and 5 jobs respectively, the load balancer directs the next available job for execution to the first node. If there are two nodes, both with queue sizes of 0, and number of running jobs 5 and 3, the load balancer sends the next job for execution to the second node. • The **Custom** type allows you to create your own unique load balancer based on your own workload distribution algorithm. If you choose Custom, you must provide **LoadBalancerClass** parameter details. Furthermore, all custom load balancing types also require a class to implement the user defined load balancing.

LoadBalancerClass	This parameter is only used if LoadBalancerType is **Custom**. Indicates the name of the user class implementing custom cluster load balancing. The user class must implement
	com.xenos.framework.system.cluster.loadbalancer.ILoadBalancer
	Example:
	`package mypackage;`
	`import com.xenos.framework.system.cluster.loadbalancer;`
	`public class MyLoader implements ILoadBalancer {`
	`/**`
	`*`
	`* @param clusterNodeList The list of all alive nodes`
	`* in the cluster. Each node is presented by string in`
	`* the following format: <IP address>:<port>`
	`* @return The address of the cluster Node where Job`
	`* will be executed. String format is the same as`
	`* input format.`
	`*/`
	`public String getNextNode(String[] clusterNodeList) {`
	`}`
	`}`

3. Create the EAR that can be used by all the nodes in a cluster using the Package and Deploy Wizard:

   Follow the Setting up a Server (Package and Deploy Wizard) instructions to create the EAR, and ensure you do the following steps:

   On the **Select the Target Application Server** screen of the **Package and Deploy Wizard**, the **Host Name** will be the host name or IP address of the host server. You can also use a variable set at runtime to minimize the number of times you need to go through the Package and Deploy Wizard. The same needs to be set for the HTTP port and JNDI port. Depending on how you define these settings, this will generate an EAR that can be used on either one node of the cluster or multiple nodes of the cluster.

On the **Configuration Manager** screen, select the **Use Repository** check box. This is essential to allow resources to be shared amongst the cluster.

4. View the cluster configuration.

   The **Connections tab** of the File Systems and Servers Window is where the servers are displayed and managed. A **cluster view** is a view of all the servers that act simultaneously to process the messages run through Output Transformation Server. Once you have set up your cluster, the ability to show a clustered view of Output Transformation Server will present itself.

   Once a connection to a clustered server has been obtained, right-click its IP address in the **Connections tab** and from the context menu that appears, select the **Show Cluster View** menu option. In the Development Window, the **Cluster View** screen appears.

Related Links

- "System Configuration Parameters" on page 17
- "System Configuration Parameters" on page 17
- "System Configuration Parameters" on page 17
- "Connections Tab" on page 279

2.2.2 Defining Your ILoadBalancer Implementation

When you set the **LoadBalancerClass** in the **ClusterManager** parameter, you need to point it to some class that implements the user defined load balance. If you do not define the field, the default LoadBalancer will be implemented.

Note: These steps only pertain to setting **Custom** load balancing types. For more information, see the content relating to the Cluster Manager and load balancing under "Setting Up Server Clustering and Load Balancing" on page 40.

To define your ILoadBalancer implementation:

1. Create a load balancing Java file and place it in the `<install_home>\initialFiles\common\sample` directory.
2. Edit your load balancing Java file according to the required logic.
3. In Output Transformation Designer, set your JDK folder to Java Home under **Tools** > **Preferences** > **Java** > **Java Setting**.
4. Compile your load balancing Java file and build a load balancing jar file.
5. Copy your load balancing jar file to `<install_home>\lib\common`.
6. Change the LoadBalancerClass in the ClusterManager parameter in `default.xSystemConfig` to `com.xenos <your Java file name>` .

7. Use the Package and Deploy wizard to build an EAR. (For more information about building an EAR, see "Setting up a Server" on page 293.)

8. Deploy your EAR and build a cluster that will run on the new balance policy.

An example of the ClusterManager parameter and the load balancing Java file, called `Testloader.java`, appears below.

2.2.2.1 TestLoader class example

```
package com.xenos.framework.system.cluster;

/**

 * Round Robin load balancer.

  */

public class DefaultLoadBalancer implements ILoadBalancer {

    private int m_lastRpcIndex=0;

    public DefaultLoadBalancer() {}

    public synchronized String getNextNode(String[] clusterNodeList){

        String result=null;

        if (m_lastRpcIndex >= clusterNodeList.length-1) {

            m_lastRpcIndex = 0;

        } else {

            m_lastRpcIndex++;

        }

        result = clusterNodeList[m_lastRpcIndex];

        return result;

    }

}
```

2.2.3 Using Failover

Failover is the continued processing of information in the event of server failure. If one server in a cluster fails or becomes unavailable due to planned downtime, the other server(s) in the cluster will assume the workload, preventing an interruption of service.

Failover is triggered by one of two possible events: master node failure or slave node failure.

- If the master node becomes unavailable, failover ensures that messages will be picked up again when the event is restarted on the new master node. (Note that for Socket and HTTP events, messages will have to be resent in order for a response to be returned to the calling client.)

- If a slave node fails, failover causes messages to be diverted to the next node in the cluster.

You can enable or disable failover at the event level using the **JobFailOver** parameter.

2.3 Logging Profiles

Logging profiles indicate to the system what types of events to log in the background while your jobs are running. Log profiles can be turned on and off by adding or removing them from the system configuration file. By default, the Debug and Fatal Error logging profiles are enabled in the system configuration.

Tip: For more information on the system configuration file, see "System Configuration" on page 17. See "Log Profile Editor tab" on page 47 for information on editing what type of information is recorded by your logging profiles.

To add/remove a log profile:

1. In Output Transformation Designer, open the **default.xSystemConfig** file.

   The system configuration file's parameters display on a new tab in the Development window.

2. Navigate through the parameters until you reach **LogProfile** > **LogProfileRef**. The value for this parameter indicates the number of logging profiles currently enabled. Click the **Ellipsis** in the **LogProfileRef value** cell or you can double-click the cell itself.

   The **Property – LogProfileRef** dialog appears.

3. On this screen, you can manage which log profiles are in use and actively logging information as your projects run. The **Available Log Profiles** pane displays all the existing log profiles on your system that you can choose from, while the **Log Profiles In Use** pane displays which logging profiles are currently active. When selecting the individual logging profiles, if available, a brief summary of the profile appears in the **Description** pane.

   You can change the status of your log profiles by switching them between the Available Log Profile and Log Profiles In Use panes by selecting a profile and moving it by clicking the arrow buttons for the appropriate direction or dragging and dropping it.

4. When you are finished selecting log profiles, click **OK**.

   The system configuration tab reappears.

5. **Save** your system configuration file.

2.3.1 Log Profile Editor tab

You can edit the log profile to set up what to log by double-clicking an existing logging profile in the File Systems window. This will open the current selected logging profile.

You can also create a new logging profile by selecting **New** from the menu bar, and selecting **System** from the component tree. The New Component Wizard assists with the process of creating the new logging profile and then the new profile is added to the File Systems window.

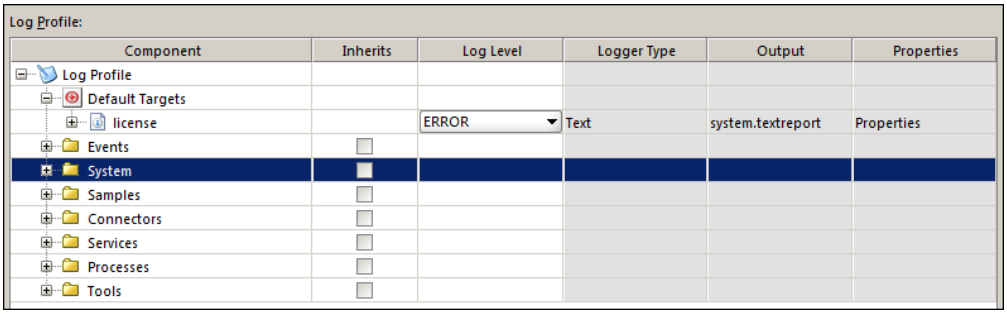


Figure 2-3: Log Profile in Development window

2.3.2 Logging targets

The profile itself is broken down into types of component groups and individual components, each having their own **Logging Targets**.

Each of the component groups and individual components has a set of flags to set that allow you to only log what is required. Individual targets can be added to each with their own set of parameter values. This allows one component to have a variety of separate log files that each contain finely configured logging information.

For more information, see "Adding Logging Targets to a Profile" on page 47.

2.3.3 Adding Logging Targets to a Profile

To add a Logging Target to the log profile of a process flow or component, do the following:

2.3.3.1 Step 1: Selecting a Process Flow or Component

1. In the **Log Profile** tree, locate and expand the process flow or component.
2. Right-click the associated Targets icon and select **Add Target**.

The figure below shows a logging target being added for the TextEbXmlProcess in the Samples folder. The logging target will be associated with this particular process.

If you were to select the Targets link associated with the Samples folder, all individual processes that sit under it can use its configurations.

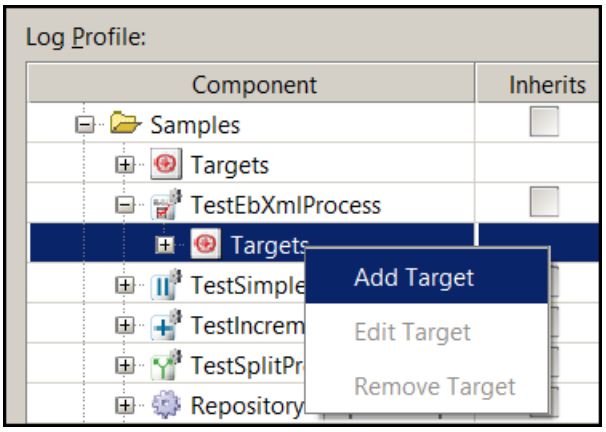


Figure 2-4: Processes in the Logging Profile tree

2.3.3.2 Step 2: Adding a Target

From the Add Target dialog window, click **New Target**,  , and add as many targets as you require.

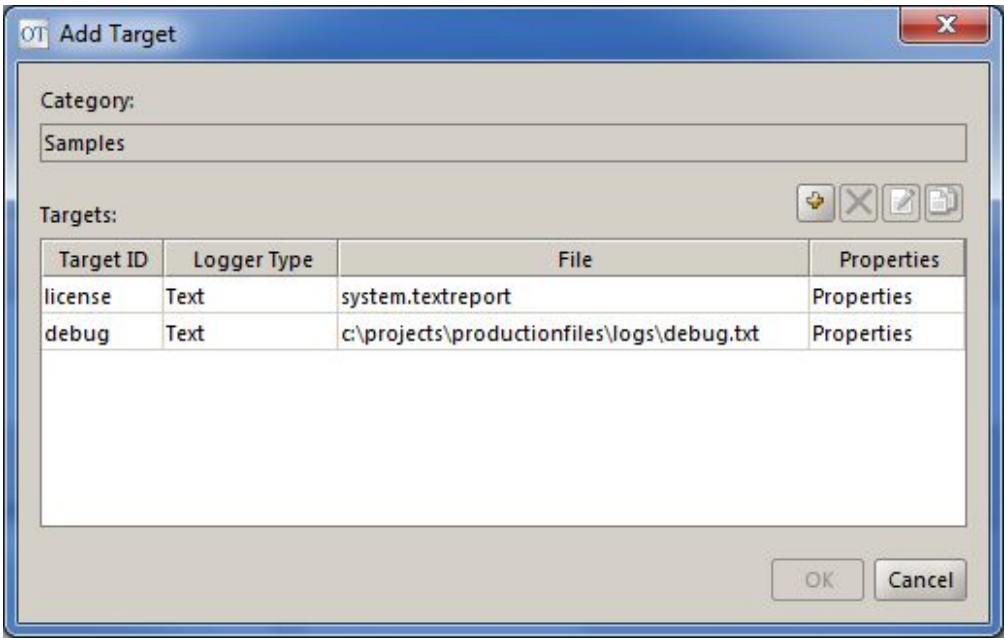


Figure 2-5: Add Target dialog

2.3.3.3 Step 3: Entering Target Properties

In the **New Target** dialog window, complete the following:

1. Set **Target Type**:

   - **Target ID**. Indicates the log file name that is used internally by Output Transformation Server. This can contain any alpha numeric characters.
   - **Logger Type**. Specifies how the log is displayed. Options are:

     – **Text**. Presents the log files in a text file.

     – **SystemOut**. Writes the log messages to the display output or another output destination specified by the host environment or user.

     – **SystemError**. Writes the log messages to the display output or another output destination specified by the host environment or user. By convention, this output stream is used to display error messages or other information that should come to the immediate attention of a user even if the principal output has been redirected to a file or other destination that is typically not continuously monitored.
   - **File**. Specifies the file name of the custom log file. The file type `.textreport` will be automatically appended to this file name.
   - **External Log File**. If the **Logger Type** is `Text`, you can check this box and enter an absolute path to where the log file will be. If this is unchecked, the output will be to a relative path based on the main mount point.

2. Set **Target Properties**:

   - **Create New File When File Reaches Size**. Indicates the maximum size of an individual log file. Enter a number and choose your desired units. This option is only available if the **Logger Type** is set to **Text**.
   - **Create New File At Specific Time**. Allows you to automatically create a new log file at predetermined intervals. Choose an interval from the drop-down menu. Older log files must be deleted manually. This option is only available if the **Logger Type** is set to **Text**.
   - **Max Backup Files**. Indicates the maximum number of backup files to save. You must specify the number of archived log files to keep once the condition for creating a new file is met. When the number of archived log files exceeds the indicated maximum, the oldest will be deleted each time a new log file is archived. If this value is set to 0, then no backup files will be created and the current log file will be deleted when a new log file is created. If the value is set to -1, then an unlimited number of archived log files can exist.
   - **Pattern**. Specifies the layout pattern composed of literal text and format expressions called “r;conversion specifiers”.

     Each conversion specifier starts with a percent sign (%) and is followed by optional format modifiers and a conversion character.

     The conversion character specifies the type of data, e.g. category, priority, date, thread name.

     The format modifiers control such things as field width, padding, and left or right justification.

     The default pattern is as follows and is explained in detail below:

     %d{yyyy-MM-dd’T’HH:mm:ss.SSS}<%t> %-5p [%c]: <%j> ${varname} %m %n

     Where:

     – **%d** specifies the date an time of the logging event.
     – **%t** specifies the name of the thread that generated the logging event.
     – **%p** specifies the priority of the logging event (info, error etc.). -5 means there will always be 5 characters reserved for this parameter
     – **%c** outputs the fully qualified class name of the caller issuing the logging request.
     – **%j** specifies the job that generated the logging event.
     – **${varname}** specifies a job variable to print out where varname is the name of the Output Transformation Server engine variable.
     – **%m** specifies the message associated with the logging event.
     – **%n** specifies the platform dependent line separator character or characters.

Click the **Ellipsis** button to choose from a list of predefined patterns, or click within the editable field to create a custom pattern.

3. Once the pattern has been set, click **OK**.

   The configured target will appear in the **Add Target** window and you will have the option to add more targets if required.

2.3.4 Functions associated with logging profiles

Functions associated with **Logging Profiles** are detailed in the table below.

Function	Description
Open	Opens the logging profile to the main window.
Close	Closes an opened and selected logging profile.
Lock	By right clicking a logging profile in the File System window, you have the ability to lock the profile and stop unauthorized changes.
Unlock	Unlocks a locked profile.
Clone	Creates a duplicate of a logging profile.
Delete	Will remove the logging profile from the file system permanently.
Rename	Allows you to rename the logging profile in the File Systems window. For more information, see *OpenText Output Transformation Designer - User Guide (VDTOTS-H-UTD)*.

2.3.5 Functions associated with individual logging targets

Functions associated with **Logging Targets** are detailed in the table below.

Function	Description
Add	Opens a dialogue window allowing the user to add and configure logging targets.
Edit	Opens a selected logging target's configuration window.
Remove	Removes the selected target from the profile.

2.4 Tuning Output Transformation Server for optimal performance

The performance of OpenText Output Transformation Server is largely dependent upon your available system resources, particularly memory and JVM heap sizes, as well as hardware resource capabilities like the network, CPU, or disk I/O speeds on your server machines. However, if you cannot augment these types of resources on your server machine then there are various ways you can calibrate Output Transformation Server for better performance when running projects. The main areas where you can revise configuration settings in order to affect performance are:

- "Modifying the Output Transformation Server system configuration settings for higher performance" on page 53
- "Modifying Event settings for higher performance" on page 55

2.4.1 Modifying Output Transformation Server configuration files

Custom configurations can be created to affect system-wide changes on either a global or individual job level, with global changes occurring in the system configuration and individual job changes occurring in project configuration files. There are two main ways of modifying the configuration files and the method you select has no bearing on the final outcome; the configuration settings just need to be deployed to the right place after you have saved your new settings.

While the outcome of the two approaches to modifying configuration files will produce the same results, there are some additional scheduling options available when making changes within Output Transformation Designer:

- When customizing the configuration files through Output Transformation Designer and deploying them to the target application server:
  - You can deploy the file to the target server with various options that define how a running Output Transformation Server engine should react upon receiving the new orders.
  - You can deploy the file to the target application server from Output Transformation Designer with different options that define when the modification will take effect for future jobs.
- When customizing the configuration files directly in the deployment directory on your target application server:
  - You can manually modify either the system or project configuration files from within the deployment directory then the changes will take effect upon restarting Output Transformation Server.

2.4.1.1 Modifying the Output Transformation Server system configuration settings for higher performance

There are several parameters in the Output Transformation Server system configuration file that can be tweaked in order to obtain enhanced levels of performance while running projects.

- **LogProfile** > **LogProfileRef**

The LogProfileRef parameter indicates to the system what logging profiles you want Output Transformation Designer to keep track of in the background while jobs are running.

By default, the Debug and Error log profiles are active in the system configuration. The largest performance gains can be achieved by disabling the Debug log profile. Since the Debug log perpetually retains a record of system and job messages plus information from the system, reports, and error logs, disabling the Debug log profile greatly reduces the amount of data that needs to be written to the disk or database, which goes a long way toward increasing performance.

For more information on managing which log profiles are in use, see "Logging Profiles" on page 46.

- **JobAuditService** > **Enabled**

The JobAuditService parameter specifies whether the job auditing services should be performed on jobs when you execute them.

When running in production, it is highly recommended that you disable the JobAuditService parameter unless you actually require auditing or querying of your jobs. Although job auditing or querying can be very effective in helping you to troubleshoot problems you may encounter while running jobs, it requires a high level of file input/output and read/write overhead so running jobs with JobAuditService enabled will impair the performance on your machine.

- **JobManager** > **MaxConcurrentJobs**

The MaxConcurrentJobs parameter indicates the maximum number of concurrent jobs that can be processed at one time.

The default setting for the maximum number of concurrent jobs is set to 5, but this value may not be suitable for all users since the ideal number is dependent on a number of different factors. You need to consider all the resources you have available to perform a single job run on the server where you are running Output Transformation Server; for instance, the memory available (RAM, JVM heap size, etc.) and your system's hardware capabilities (disk I/O, network, and CPU speeds, number of processors, etc.). Nonetheless, it is important to note that assigning a higher value to the MaxConcurrentJobs parameter does not necessarily mean that the same number of jobs can be processed in a shorter period of time.

To determine the best value for MaxConcurrentJobs for your particular machine, you can try the following methodology:

1. In your system configuration file, set the **MaxConcurrentJobs** parameter to **1** in order to run a single job. Save the settings and run a project.
2. Using Windows Task Manager or your preferred performance metrics application, observe how much memory and CPU usage is used to complete the single job run.
3. Gradually increase the value of MaxConcurrentJobs in the system configuration file to allow multiple jobs to be processed concurrently until it reaches an optimal stage where the JVM does not run out of memory and a level of relatively high CPU usage can be sustained.

- **XDocProperties**

The XDocProperties and its child parameters together define the properties for XDocs across the entire system.

The InputBuffer and ResultBuffer parameters indicate the size of the buffer available for reading input and writing output data, respectively. The larger the values are for these parameters, the more memory that is available to Output Transformation Server to temporarily store data content in the buffer during file input and output activities, which results in faster job processing times.

The SpillThreshold parameter specifies the amount of data that can be stored in memory before writing to a temporary file. The data in the temporary file is then automatically read back to memory when it is needed. The bigger the SpillThreshold value is, the faster a job can be processed because more memory is devoted to storing data in the buffer before writing the data content to the disk as a temporary file.

The CompressThreshold parameters dictate the upper limit size for XDocs after which they will be compressed. While the act of compression itself does not severely impact performance, setting the threshold at too low a value for your project's requirements will send more files through the compression process which will ultimately degrade the overall performance of Output Transformation Server.

- **ComponentPoolManager**

The ComponentPoolManager and its child parameters help to define how a job should be handled when within the context of a specific runnable component.

After setting up the ComponentPoolManager, each component defined in your project can be pooled by the engine to improve performance through streamlining of the load process to include only the necessary components for your project.

When deploying projects to your target application server from Output Transformation Designer, the required components are automatically added to the component pool.

2.4.1.2 Modifying Event settings for higher performance

For Event configurations, there is only one parameter that can be tweaked in order to extract higher levels of performance while running projects: **MaxSubmittedJobs**. The MaxSubmittedJobs parameter defines the maximum number of job requests that can be handled by the event during a single processing cycle. If any subsequent requests are sent, they must wait for another one to finish before being processed.

By setting a higher value for the MaxSubmittedJobs parameter, more jobs can be submitted to the Output Transformation Server engine and initialized for processing, which will greatly minimize the length of time needed to complete a project run since multiple jobs are processed concurrently.

Tip: If MaxSubmittedJobs contains a value that is greater than the value defined for MaxConcurrentJobs in your system configuration, then the MaxConcurrentJobs value takes precedence and only this number of jobs is processed simultaneously by the Output Transformation Server engine. The remainder of the submitted jobs are queued and processed on a first come, first served basis when a previous job has completed processing.

2.5 System properties

In the `<install_home>\settings` folder, there are several properties files that users can modify in order to establish system-wide settings. Setting these values incorrectly can have adverse effects on your installation, so you must be certain when editing these files. Many of these files contain some brief comments on their functionality.

2.5.1 Setting a separate work directory

By default, your work directory is kept inside the installation directory. This is suitable for most situations when there is only one user, but administrators with multiple users may want to provide each user with their own workspace area to minimize the possibility of conflicts with project settings. If you want to keep your installation directory separate from your working directory, then you can follow these steps:

Note: : Your base installation folder and subsequent references to it will be represented by the `<OTS_home>` variable while the work area folder is represented by the `<OTS_workarea>` variable.

1. Navigate to the *`<OTS_home>`*`\settings\` folder and open the `SetOtsWorkarea.bat` file for editing.
2. Either delete or comment out the following two lines:

```
set OTS_WORKAREA=
goto END
```

3. To define the new working directory, locate the set `OTS_WORKAREA=%UserProfile%\.ots\workarea_16.5` line and replace the existing value with the desired working directory. If the work area folder you specify does not already exist, it will automatically be created. (By default, `%UserProfile%` is a system variable that indicates where a user's home directory is located, which is set to the `C:\Users` directory for Windows users.) When you are finished, **save** your work.

4. Navigate to the *<OTS_home>*`\startup` folder and launch the application by running the `startDesigner.bat` file.

While Output Transformation Designer is starting, the new work area is initialized. Subsequent launches of the application by this user can use either the `startDesigner.bat` or `startDesigner.exe` files under `<OTS_home>\startup` or the user's own `<OTS_workarea>\startup` folder since the work area is switched automatically for the particular user. Also, bear in mind that any users running the program will hold full permissions to the working directory.

2.5.2 Setting the Java version

To quickly establish the JAVA_HOME property to be used by Output Transformation Designer, the Output Transformation Server engine, and your application servers, you can set this property in the `SetJavaHome.bat/.sh` file located in the `<install_home>\settings` directory. (This file is only created after the application has been started for the first time.) If no value is set for `OT_JAVA_HOME=` in this file, then the default JDK or JRE from the PATH environment variable is used instead.

You can verify the Java version the application is currently using, by navigating to **Help** > **About** and switching to the **VM Info** tab where you can review your **java.home** setting.

2.5.3 Maintaining separate properties for multiple instances

A custom properties file is created for each instance of Output Transformation Designer or the engine that is started. Created during start up, these properties files are used to populate the system properties and applies to all start up modes, including designer, appserver, and standalone engine modes. By maintaining individual properties files for each instance, administration of multiple instances is simplified since numerous properties can be defined in an easily accessible spot. All properties files are created in the `<install_home>\settings` directory and follow the `<instance_type>_<instance_name>.properties` naming convention. For example, `designer_local.properties` or `tomcat_ots.properties` are valid instance properties file names.

Within the file name, the `<instance_name>` must correspond with the settings stored inside the `startDesigner.bat/.sh` file. If there is a discrepancy between the instance names, it is considered as a new instance. For example, if you are starting a new instance of designer mode, you must ensure that the instance name shown in the `designer_<instance_name>.properties` file name and the `-instance=<instance_name>` property within `startDesigner.bat/.sh` are consistent.

Moreover, when the application is started in any mode, the `<install_home>\settings\startup.properties` is automatically updated with any arguments set within the instance property files. This file can contain the details of multiple instances, with each argument delimited by the instance name. As an illustration, assume a user has set up two separate instances of designer mode and has named them **Designer** and **Production**. Each of these instances will have their own start up files containing their instance names, `startDesigner.bat` and `startDesigner(Production).bat` in this case, stored in the `startup` directory. (Their respective instance property files are created upon starting them up for the first time.) Within these files, the `-instance=` property can be located and for the Production instance in this example, is:

```
-runmode=designer -instance=Production %1 %2 %3
```

After starting up the two instances, if you open the `startup.properties` file to review the settings then you will see new arguments that start with the instance name it pertains to. Some sample entries may be:

```
Production.jvmarg=-Xmx800M -Xms300M
Production.version=16.7.DEV.00_1234_190101
Production.name=Production
```

When establishing properties within the instance properties file, you can use any property that is present in the `default.xSystemConfig` file. You can also use new properties that are unique to each instance and are not shared across multiple instances. If you want to use a new property across multiple property files, you must duplicate the property across all the individual property files. Following edits being made to any properties file, you must restart the instance to reload the updated properties before any of the changes can take effect.

2.6 Running JavaScript

Consider the following guidelines when using JavaScript in your projects.

2.6.1 Maintaining JavaScript Security

Vulnerabilities in software can result in compromises in your security posture, which can circumvent the safety measures set up to protect your organization from both internal and external threats. While the probability of a breach is highly unlikely, it is important to be aware that the risk still exists and the damage caused by a malicious script can be critical, due to either the crashing of a server or the loss of sensitive data. The software supports JavaScript and unauthorized users may attempt to gain access by executing malicious JavaScript so it is imperative to secure your system to safeguard against these types of attacks.

Within all products in the Output Transformation Suite, a `JavaScriptPermissions.properties` file is used to authorize which parameters can be used to run `@js` and `@jsFile` JavaScript functions. This file is stored in your `<install_home>\BaseRepositories\<appserver>\<instance_name>\common\_configs` directory. By default, no parameters are allowed to run `@js` functions while all parameters are allowed to run `@jsFile` functions.

Note: The `@jsFile` function runs a JavaScript located on the server's file system and is wholly controlled by administrators, thus, it does not pose the same threat as `@js` functions.

When opening a project that includes a property containing some JavaScript, the **Security** dialog appears and prompts you to decide how to handle execution of the script since permission must be explicitly granted in order for the script to run. The following options are available:

- **Keep blocking**. Continues to block the script and does not allow it to run.
- **Allow during this session**. Permits the script to run for the duration of the current session. Upon shutting down your VM session and then later reconnecting to the VM, you will be asked again whether the script is allowed to run.
- **Always allow**. Permanently allows the script to run. Users can go into the `JavaScriptPermissions.properties` file to manually add any scripts that are permitted to run, however, selecting the Always allow option performs the same function.

2.6.2 JavaScript library compatibility

This application provides JavaScript customization for dynamic configuration parameter evaluation and custom component logic. Beginning with Java version 15, JVM installations discontinued providing a JavaScript engine by default. Starting with Output Transformation Server version 22.4, the Nashorn JavaScript library that was previously included with the JVM is now shipped as a third-party JAR library with the product. Including the Nashorn library provides seamless support for users who upgrade their product to version 22.4 and their Java to version 17 or later.

Chapter 3

File Systems

The **File System** is where your resources and your configurations are stored. File systems can be local or network directories, compressed files, either ZIP or JAR, relational databases, or content storage systems.

The File Systems window is where the file systems are displayed and managed. For more information, see *OpenText Output Transformation Designer - User Guide (VDTOTS-H-UTD)*.

3.1 File Systems Tab

To view and manage your mounted file systems, click on the **File Systems** tab, located at the top of the File Systems window.

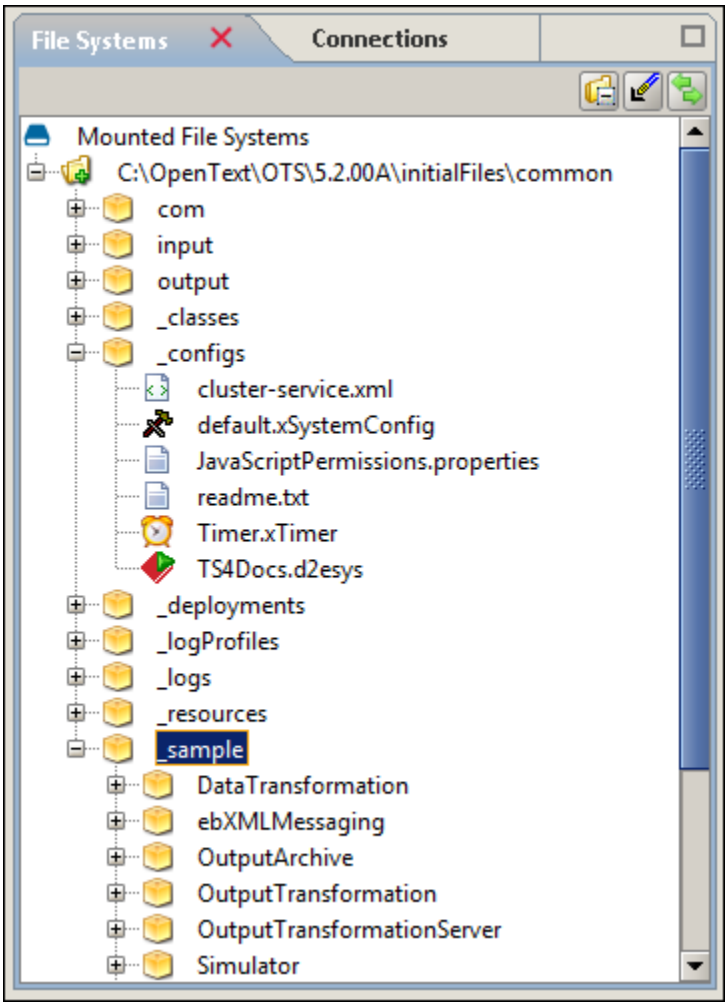


Figure 3-1: File Systems tab

In addition to displaying your mounted file systems, the **File Systems** tab also indicates the available resources associated with each file system, categorized under the following folders:

com	Provides sample Java code for the implementation of custom components.
_classes	Contains `.class` files, and is populated by the OpenText Developer IDE (Integrated Development Environment). See *OpenText Output Transformation Server Developer's Guide* for more information.
_configs	Contains the system configuration files.
_deployments	Stores project deployments.

_logProfiles	Stores logging profiles.
_logs	The location where all OpenText engine logs are written.
_resources	Contains all resources for events and other components.
_sample	Provides sample project files for each OpenText plug-in that is installed. Sample event and process flow files are found under: `<install_home>\initialFiles\common\_sample\OutputTransformationServer\`
_stats	The location where historical statistics are written.

3.2 Mounting a File System

You must mount a file system in order to work with its resources and configurations in Output Transformation Designer. To mount a file system:

1. In Output Transformation Designer, right-click anywhere in the **File Systems** window.

   The File Systems context menu appears.

2. In the context menu, hover over the **Mount** option. The menu branches out where you must select the type of file system you wish to mount, either **Local Directory** or **Archive**. A local directory is a folder on your local machine that contains your resources and configurations while an archive contains your resources and configurations in a compressed file.

   The **Mount File System** dialog appears.

3. The Mount File System dialog contains a **Local** tab and an **Archive** tab. Depending on the type of file system you selected in the previous step, the respective tab appears. (If you change your mind about the type of file system you want to mount, you can switch tabs to the other type.)

   If you are on the **Local** tab, navigate to the file path location that contains your resources and configurations and select the folder you wish to use as your mount point; all files and folders contained hierarchically below the mount point are also included in the file system. The mount point location can also be typed into the **Folder Name** field. Once you have finished making your selection, click **Mount**.

   If you are on the **Archive** tab, navigate to the file path location that contains your archive file, which can only be a ZIP or JAR file. The archive file's path location can also be typed into the **File Name** field. Once you have finished making your selection, click **Mount**.

   The **Building Environment** dialog appears.

4. The Building Environment displays Output Transformation Designer's progress as it builds the file system. It automatically closes when the operation is complete and your mounted file system appears on the **File Systems** window.

3.3 File System Functions

The following table details functions available that affect file systems in the File Systems window.

Function	Description
Mount	Mounts a file system of the following types: • Local directory • Compressed file
Unmount	Unmounts the selected file system.
Refresh	Refreshes the file system directory view.
Deploy	Deploys a selected file system to a connected server.

Chapter 4

Process Flows

Processes are runnable components that take zero or more inputs and generate zero or more results. An example of a process is one that may write an output file, send an email or JMS message. OpenText Embedded Data Transformation Engine is an example of a process which can be invoked in a process flow to convert one data format to another.

A process flow executes a series of processes, in a specific order. A process flow can contain processes or other process flows. Process flows are reusable and can be triggered by events or through the RunJob API. The process flow name normally gives a shorthand description of what the flow does, and uses an extension of `xProcessFlow` in the file name. Several samples of process flows are provided in the samples directory:

```
<install_home>\initialFiles\common\_sample\OutputTransformationServer\
processFlow
```

Each event triggers a process flow, as specified in the process flow's ProcessToRun parameter. A process flow is triggered when an event is invoked by its specific functionality. For example, the file event will invoke a process flow when a file is dropped into the specified directory.

The process flow parameters (shown in the Properties Window) include a list of processes to be performed, and parameters for each process. There is no need to edit this list – it is built automatically by dragging processes into the flow chart. Simply set the parameters for each individual process from the flow diagram, including the error process. For more information, see "Adding Components to Process Flows" on page 68.

A process flow has a start point that is linked to the first process, and an end point that is linked to from the last process. A parameter can also be linked to the Catch area of the Process Flow tab should the need arise for you to run your own process on the event of an exception occurring.

4.1 Process Flow Tab

When a process flow is initially opened in the **Development window**, you see a screen similar to the one below. The initial process flow contains a starting point, an end point, and an icon that relates to error catching. (For more information on error catching, see "Catch onError" on page 69.)

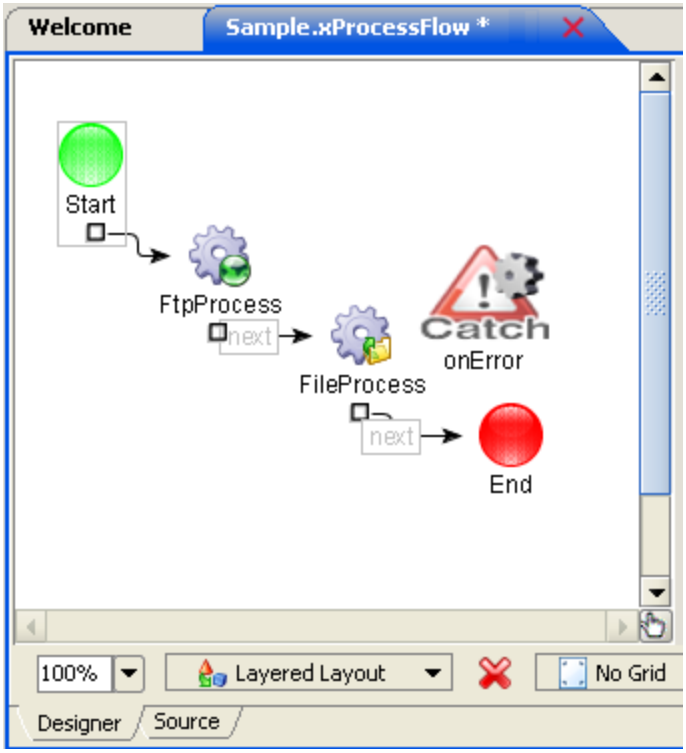


Figure 4-1: Process Flow tab in Development window

The following figure shows a configured Process Flow.

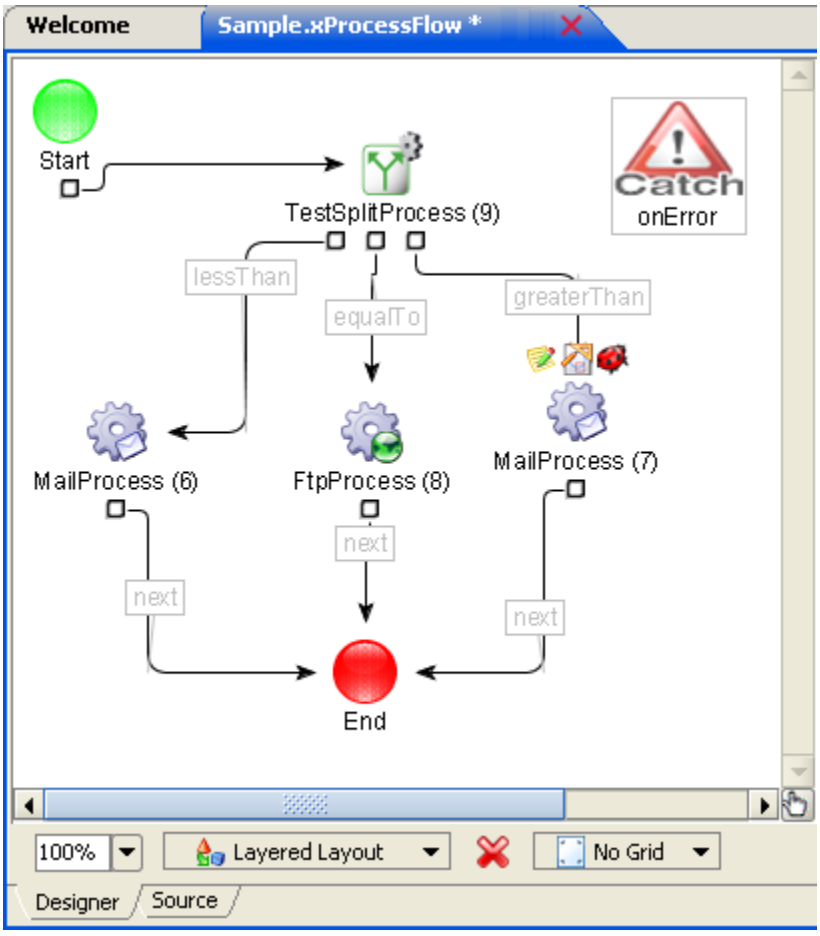


Figure 4-2: Configured Process Flow

4.1.1 Process Flow Designer Tab

The **Designer** tab, located in the bottom left corner of the **Process Flow** tab, displays a graphical representation of the process flow. It shows what processes will be run, and in what order they will be run. It is also where you build the process flow by adding components and connecting them together. The flow starts at the green circle and ends at the red circle. Required processes are dragged into the flow from the **Component Palette** on the right side. They can then be connected with arrows to indicate the order of execution. For more information, see "Adding Components to Process Flows" on page 68.

4.1.2 Process Flow Source Tab

As the process is built up, the XML source code for the process flow is created behind the scenes. To see the XML source code, switch to the Source tab, located in the bottom left corner of the Process Flow tab, and you are presented with an XML representation of the currently displayed process flow.

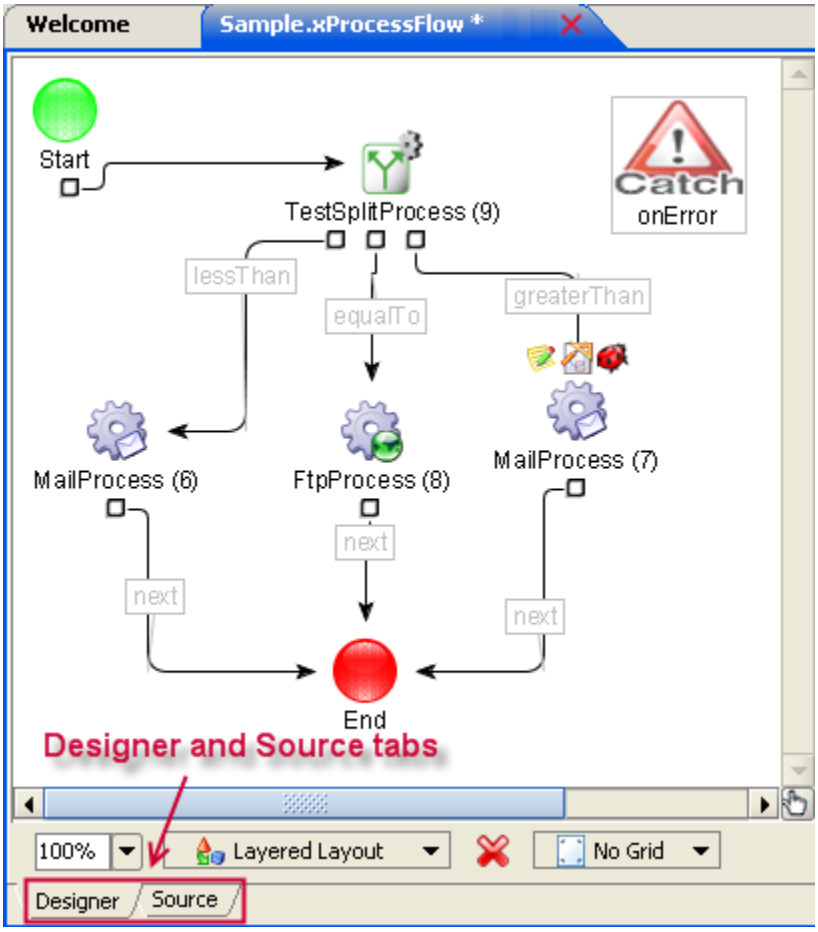


Figure 4-3: Designer and Source tabs

4.2 Process Flow Palette

To the right of the Development window is the Process Flow Palette. The palette displays processes and process flow components. These components can be dragged into the Development window and directly into process flows in whatever order is required.

The components are grouped together by category and their placement in the palette is a representation of their placement in your file system.

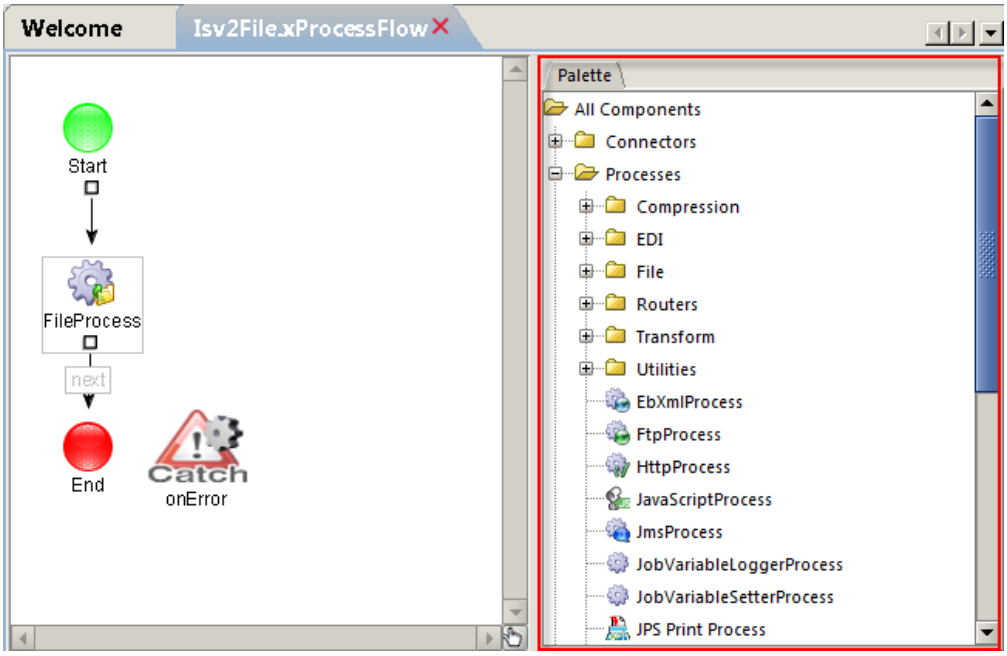


Figure 4-4: Process Flow Palette tab

4.3 Setting Up Process Flows

The topics in this section describe setting up a process flow.

4.3.1 Setting Up a New Process Flow

A process flow links various processes together and executes them in a specific order. To configure a new process flow:

1. Start the New Component Wizard by either clicking the **New** icon,  , on the toolbar or by navigating to **File** > **New**.

   The **Select a File Type** screen of the **New Component Wizard** displays.

2. Expand the tree under the **Processes folder and select ProcessFlow**, then click **Next**.

   The **Name and Location of New ProcessFlow** screen appears.

3. The following information must be completed:

   - **Name**. Specifies the name for the process flow. Use a name that will help you to remember what it does to make it easier for yourself or others to find the next time you need to use the same set of processes. You do not need to include the extension as it will automatically be given the extension `.xProcessFlow` in the file name.
   - **File System**. Indicates the file system in which to save the process flow; this will indicate the base directory to store the created process flow.
   - **Directory**. Indicates the specific directory for your process flow. You can choose a directory that already exists in the file system or create a new one.

   When you are finished, click **Finish**.

   The new process flow appears on a separate tab in the Development window. Your new process flow only contains a Start icon, an End icon, and a warning sign for Catch on Error. You can customize your process flow by adding components to it. For more information on adding components, see "Adding Components to Process Flows" on page 68.

4.3.2 Adding Components to Process Flows

To build a process flow:

1. Drag components from one of two areas of Output Transformation Designer:

   - Component Palette
   - File Systems tab, when pre-configured components have already been added to a file system.

   The components are dragged to the **Development window** of the process flow and linked to other processes, start points, and end points.

   As you add each component, the **Add Component window** appears.

2. Select one of the following options:

- **Single Use**. If the process is only to be used one time, and the configuration will not be saved to a separate file.
- **Shared**. If you will need the same process configuration in other process flows, and you will need to enter a name and directory where this configuration will be stored on disk. You might do this for your email configuration, if several different process flows may need to access the same account to send email messages.
- **Existing**. If the process configuration already exists, such as an `.mgp` file, then select the desired configuration from the list displayed in the box at the bottom of the pane.

3. Click **OK**.

   The Add Component dialog closes and your component is added to the process flow.

4.3.3 Catch onError

Catch onError is used to indicate what action should take place when an error occurs during the running of your process flow. It does not need to be attached to the flow with connection arrows; Catch OnError just needs to be configured properly to be incorporated into your process flow as the feature is available in all process flows.

The most common use of Catch onError is in conjunction with Mail Processes in order to receive an email when errors occur. You can use job variables like ${JobTicket.fatals} or ${JobTicket.warnings} to customize the types of errors you want to receive notification of. For more information on job variables, see "Using Error Variables" on page 339.

Figure 4-5: Catch onError icon

4.3.3.1 Assigning an Error Process

Any type of process can be assigned to Catch onError, thereby having a process automatically initialize upon the occurrence of an error in your process flow.

4.3.3.1.1 Assigning an Error Process to a Pre-configured Flow

To assign an error process to a pre-configured process flow:

1. Locate the **Catch onError** icon in your process flow.
2. You have two options to designate the Catch onError process flow:
3. From the **File Systems** window, navigate to the directory containing your process, then drag the process and drop it onto the **Catch onError** icon.
4. From the **Palette**, in the **Processes** folder, locate the ProcessFlow icon, then drag and drop it onto the **Catch onError** icon. The **Add Component** dialog appears. Click on **Existing**, and from the table that shows all the available process flows, select the one you want to appoint as the Catch onError process, and click **OK**.

The Catch onError icon changes to show a gear wheel in the top right corner to indicate that a process has been assigned. To view or edit the properties, double click on the icon, and the process flow opens in a new tab in the Development Window.

Figure 4-6: Catch onError icon with a Process Assigned

4.3.3.1.2 Assigning an Error Process to a Component or Empty Process Flow

To assign an error process to a process component or an empty process flow:

1. Locate the **Catch onError** icon in your process flow.
2. Drag a **process component** or **empty process flow** from the **Palette**, and drop it onto the Catch onError icon.
3. You must select whether this process will be **Single Use** or **Shared**. Once you are finished making your selections, click **OK**.

The Catch onError icon changes to show a gear wheel in the top right corner to indicate that a process has been assigned.

Note: If you are using either new or shared processes, you must remember to configure their properties. To edit the process' parameters, double click on the

Catch onError icon and a dialog appears containing the properties available to be set. For process components that have an available wizard, the Wizard appears instead. Once you are finished setting up your process, click **OK**.

4.3.3.2 Removing an Error Process

To remove an error process:

- Right-click on the **Catch onError** icon, and from the context menu that appears, select **Remove**.

The gear wheel disappears to indicate that your error process is removed.

4.3.4 Preventing XDoc Creation for Large Jobs

When running a transform where the project (custom IOHandler, etc.) itself creates a large file on the file system users may wish to prevent an alternate xdoc from being created.

To bypass XDoc creation:

1. Right-click the Data Transformation Engine or Output Transformation Engine component in the process flow and choose **Show Mappings** from the context menu.
2. Click the **Result XDocs** tab.
3. Click the **Output button**  to open the **Mapping Details** dialog.

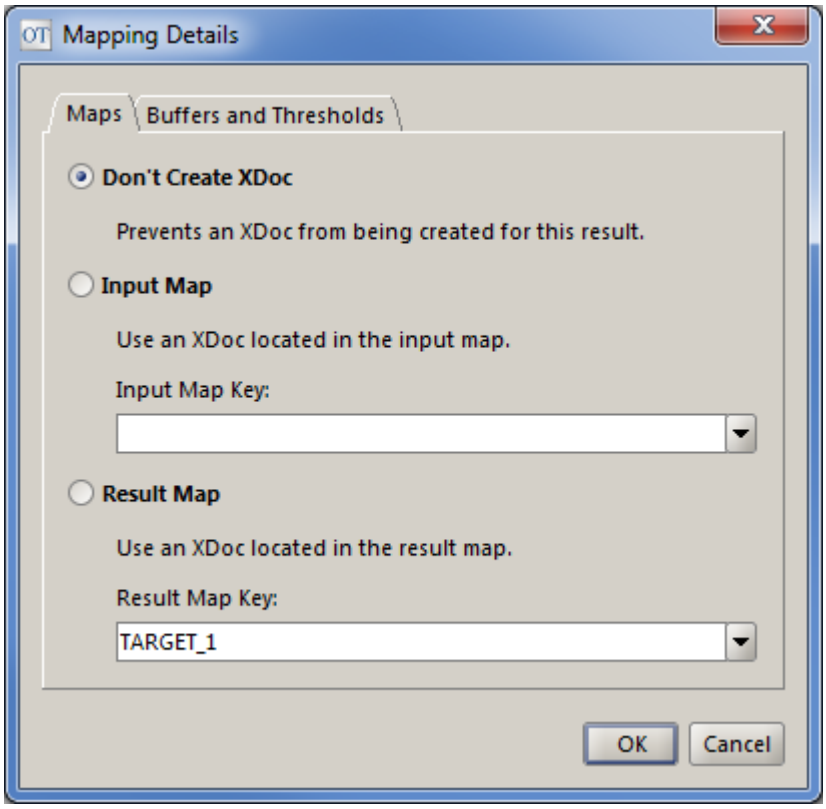


Figure 4-7: Mapping Details screen

4. Choose **Don't Create XDoc**.

5. Click **OK** to close the dialog and return to the Source and Results Mapping dialog.

   The **Destination** field will be empty and the icon changed to  .

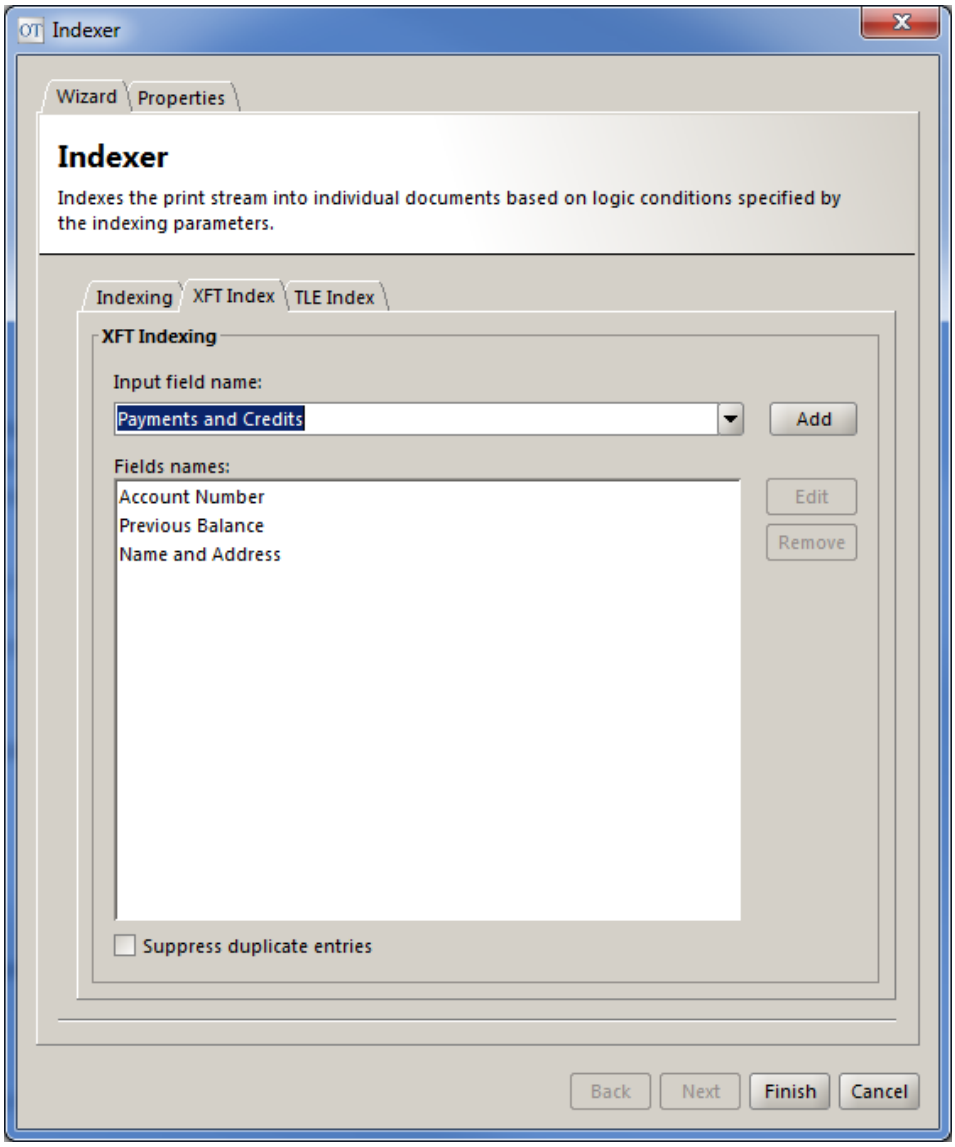
   If you preview the process flow source code, the **Destination** field from the form maps to the **nameToUse** field, which will now be empty ("").

```
<resultMappings name="visiontest.ComponentList.Index Writer.Parms.OnDemandOptions.OutputFile"
                nameToUse=""
                destination="resultMap">
```

4.3.5 Using XDoc Index Information in Process Flows

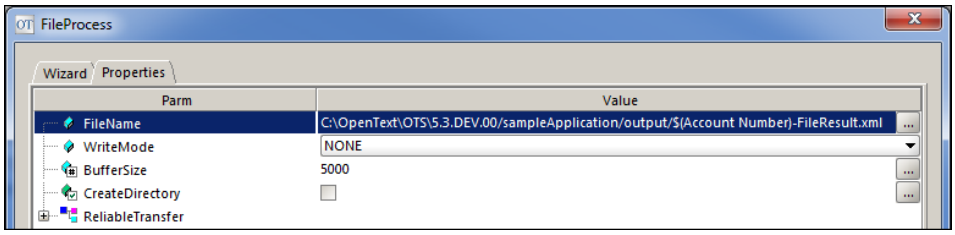
Index information can be collected within Output Transformation Project and set using the **XDocIoHandlerFactory**. Once a single result is pulled out of the **MultiXDocSplitter**, the corresponding information can then be used in Output Transformation Server process flows as variables. These variables can be used within any other component including the FileProcess and the JobVariableRouter.

1. Set the option to collect index information (properties) in the **Output Transformation Engine Properties** by setting the **CollectIndexerInfoOptions** parameter to **Enabled**.
2. Ensure there is an **Indexer** component in the Output Transformation Engine project.
3. Configure the **Indexer** component with the fields you want to index. You can enable one or multiple types of indexing, including XFT, TLE, and comment indexing.



4. In your Output Transformation Server process flow, you can now reference these indexes and use them as variables in process flow components such as routers and processes.

   For example, in this FileProcess, the variable name (Account Number) is entered within the FileName in the component properties.



Related Links

- "Output Transformation Project" on page 209
- "MultiXDocSplitter" on page 206
- "FileProcess" on page 198
- "JobVariableRouter" on page 201
- "JobVariableRouter" on page 201

4.3.6 Using Completion Code Information in Process Flows

A **completion code** is an attribute that an Output Transformation Project component in a Output Transformation Server process flow stores in Output Transformation Engine's JobRecord while the Output Transformation Engine project in the process flow runs. The JobRecord stores the completion code's value and description in two job variables: **d2e.completion.id** and **d2e.completion.desc**. The JobRecord lives as long as the Output Transformation Engine project in the Output Transformation Project component is processing information. Once the job finishes, the completion code value and description are sent to the JobTicket, where the Output Transformation Server process flow uses and validates the information in other processes.

There are several possible conditions that may cause a Output Transformation Engine project job to finish, including it:

- Ran successfully without any errors.
- Failed to initialize due to a configuration error.

The results appear in the Job Log tab of the **Events** Window for you to review.

Your business logic may require you to identify the completion code and take action depending on the actual value of the completion code. For example, you can use the JobVariableRouter component to create many branches that deal with each error or warning type. One branch would deal simply with successful routing. For more information, see "JobVariableRouter" on page 201

To use a completion code to validate information:

1. Create a process flow that has an Output Transformation Project component. It should include a Output Transformation Engine project inside it.

2. Add to the process flow at least one custom component or a router, such as the JobVariableRouter. These components need to come after the Output Transformation Project component in the process flow.

3. Add a string inside the custom component or router that will retrieve a completion code's value and description from the JobRecord and move it into the JobTicket.

   The following strings are examples for retrieving the completion code's value and description from a custom component.

   - d2e.completion.id:

     ```
     JobTicket.getJobVarAsString(RunVision.d2e.completion.id)
     ```

   - d2e.completion.desc:

     ```
     JobTicket.getJobVarAsString(RunVision.d2e.completion.desc)
     ```

4. Run the process flow. After the Output Transformation Project component ends, the completion code's information moves to the JobTicket, where Output Transformation Server uses it further in the process flow in other components.

5. Retrieve the completion code from the Job Log tab in the **Events** Window. Here is an example of a completion code message in the Job Log tab.

   ```
   2010-02-16T10:47:47.872-05:00 <PDF Generator-writer-0> INFO  :
   CompletionCode: Value=4, Description=The task has completed successfully,
   but one or more warning conditions have been met.
   ```

Table 4-1: Completion Code Values

d2e.completion.id	d2e.completion.desc
0	The task has completed successfully without errors.
4	The task has completed successfully, but one or more warning conditions have been met.
8	The task has failed because one or more error conditions have been met.
12	The task has failed because one or more unexpected runtime exceptions have occurred.
16	The task has failed due to an initialization error.
20	The task has been ended prematurely by request.
24	The system has denied the execution of this task.
28	The system has denied the execution of this task because there are not enough resources available.

For more information about the Output Transformation Project component, see "Output Transformation Project" on page 209.

4.4 Running Process Flows

The topics in this section describe running process flows.

4.4.1 Executing a Process Flow

After you have fully configured a process flow with all of your required components, you can run the process flow by performing one of the following actions:

- Click **Execute**,  , on the toolbar.
- From the **File Systems** tab, right-click the project and select **Execute**.
- Select **Build** > **Execute**.

When a process flow is executed, it is automatically saved if there are any unsaved changes to the file and the **Execution Options** dialog box opens, where you can provide some further job variable and source settings for your components. After your options are configured, click **Execute** to run the project.

When the project is running, the results appear in the **Output** view, which opens in the **Events** window.

Note: You can use the **Undo** function to revert changes to the saved process flow.

4.4.2 Running a Sample Process Flow

To run a sample process flow:

1. In the **File Systems** tab, expand the subdirectories under `<install_home>\initialFiles\common` to show the files under `\_sample\OutputTransformationServer\processFlow`
2. Right-click on a process flow, and select **Execute** from the menu.

   The **Execution Options** dialog is displayed:
3. Enter details for job variables, sources, and results, using their respective tabs, and click **Execute**.
4. The process flow begins running.

   The execution results are displayed in the **Output - <process that was run>** tab of the Events window.

4.4.3 Debugging a Process Flow

The Debug feature allows you to quickly diagnose problems in a process flow.

To start the debugger:

1. In the **File System** window, open the process flow to be debugged.

   The process flow opens in the **Development** window in a new tab.

2. In the Development window, hover over a process and set a breakpoint by clicking the **Debug** icon,  . Setting a breakpoint identifies where the debugger will pause execution.

   A **Pause** icon,  , appears over the selected process.

3. In the **Build** toolbar, click the **Debug** icon,  , to start the debugger.

   This opens the **Execution Options** dialog.

4. In the **Execution Options** dialog, supply all job variables and source documents needed in the process flow. In most cases, these would be the values that an Event would supply to the process flow.

5. When complete, click **Execute**.

   The **Output** and **Debugger** views appear in the Events window, and the debugging process starts.

6. Execution will pause on the process where the first breakpoint was set.

   A green arrow will appear over that process to indicate the current point of execution. The Debugger view in the Events Window will also show the current point of execution in the **Stack** tab.

7. Clicking the **Step Over**,  , tool icon will move to the next process. If the current point of execution is another process flow, then pressing **Step In**,  , will open that process flow in another editor and start debugging it as well. Clicking **Step Out**,  , while in the nested process flow will complete its execution and go back to the outer process flow. Press the **Stop**,  , tool button to stop all debugger operations.

8. You can inspect the runtime properties using the **Output** and **Debugger** views located in the Events Window. For more information on the **Output** view see "Reviewing the Output tab" on page 79.

   The **Debugger** tab is only shown when debugging a process flow. This view shows the process stack, a list of variables, and breakpoints used in the job. In the **Stack** tab, double-click a process to select that process in an editor.

4.4.4 Reviewing the Output tab

When a process flow is run, an **Output** tab corresponding to the project opens in the **Events pane**. The Output tab displays information about the output file being generated, and stores messages written during the operations conducted by in the debug logs. These operations messages include activities related to the loading, running, and saving of projects, as well as XDoc usage.

There are four sub-tabs on the **Output** tab:

- Job Stats
- Log
- Variables
- Results

4.4.4.1 Job Stats tab

The **Job Stats** tab displays details about the job, including its current status, job ticket ID, processing time, and any messages, warnings, or errors that occurred during processing.

At the top of the tab, the **Stop a running job** button,  , appears. You can click this button to cancel an executed job that appears to be hanging or running for an unusually long time. This allows you to avoid restarting the engine to remove the job.

The Job Stats table displays the following statistics:

- **Current Status**. Indicates the present standing of your job. There are three categories:
  - **Running**. Indicates that the job is currently running.
  - **Finished**. Indicates that the job is no longer running. This does not mean that there were no errors, but rather that the job did not fail.
  - **Errored**. Indicates that the job was unable to run to completion due to an error. There will be a corresponding entry at **Failure Message** to help with identifying the cause of the error.

   **Note:** The way Reliable Transfer is configured for components within the process flow will influence how errors are reported and displayed. For more information, see "ReliableTransfer" on page 278.
- **Job Ticket ID**. Specifies the unique ID assigned for the execution of this job. The format is `YYYYMMDD_HHMM_IncrementalNumber`. (Note that the HHMM time variable uses a 24 hour clock.)
- **Processing Time (ms)**. Specifies the total time, in milliseconds, that elapsed while processing the job.

- **Input Size (bytes)**. Specifies the total size of the input file, in bytes.
- **Output Size (bytes)**. Specifies the total size of the output file, in bytes.
- **Invocation Method**. Specifies how the process flow was initiated. There is also a property in the Logging Profile to track how the process flow was executed. This field will be filled in when the process flow has been executed by a file event or another process flow.
- **Failure Message**. Specifies the failure message if the result of the job is Errored. Either double-click the field entry or click the ellipsis to see the log file detailing the error.
- **Warnings**. Indicates the number of warnings from your current job. A warning usually triggers a non-fatal error. For example, if a job is configured to write to a specific file that already exists and the job does not have permission to overwrite, the warning would be that the file already exists and the non-fatal error would be that the job did not complete successfully. Either double-click the field entry or click the ellipsis button to see the log file detailing the error.
- **Non-Fatal Errors**. Indicates the number of non-fatal errors from your current job. A non-fatal error is an error condition that may or may not fail a job. Either double-click the field entry or click the ellipsis button to see the log file detailing the error.
- **Fatal errors**. Indicates the number of fatal errors from your current job. Fatal errors are absolute error conditions where the job fails. This can be caused by things like a component unable to initialize, a job not able to be executed, etc. Either double-click the field entry or click the ellipsis button to see the log file detailing the error.

4.4.4.2 Log tab

The **Log** tab displays the log for the process flow that was run.

You can also create custom log files where this information will be written. For more information, see "Adding Logging Targets to a Profile" on page 47.

The following information is displayed on the Log tab:

-  , **Hide Java exception details**. Determines whether details of Java exceptions are displayed. This function is only active for certain components.
-  , **Prevents automatic scrolling**. Determines whether automatic scrolling during population of the log is enabled. If enabled, the default view is the beginning, or top, of the log whereas usually, the default view is the end, or bottom, of the log.
-  , **Print**. Sends the log to a printer for printing.
-  , **Save**. Saves the log to a specified file within the file system.
-  , **Clear**. Clears the contents of the Log tab.
- **Search**. Conducts a case-insensitive search of the messages in the Log tab. Enter your search criteria into the **Search** field and click **Find**,  . If there are any

matching search results, the first instance of a matching string is highlighted. If you have more than one search result, subsequent matches can be located by using the **Next**, , and **Previous**, , buttons.

Log levels

The **Log Level** list is used to filter the log messages by type. The following options are available:

- **Debug**. Displays error messages of all types: DEBUG, INFO, WARN, ERROR, and FATAL.
- **Info**. Displays all error messages except for messages designated as Debug.
- **Warning**. Displays all error messages except for messages designated as Debug and Info. These messages are issued when a situation arises that may affect your final results.
- **Error**. Displays all error messages except for messages designated as Debug, Info, and Warning. These messages indicate a fault that may still produce an output, but that the output file is unsatisfactory due to these errors.
- **Fatal**. Displays only those messages designated as FATAL, which are typically critical error messages.

4.4.4.3 Variables tab

The **Variables** tab displays a list of variables used in your project, including their values and scope, following the execution of a process flow. You can consult this list to troubleshoot issues you are having that may be caused by variables.

The table on the Variables tab contains the following information about the variables used in your process flow:

- **Name**. Specifies the name used to identify the job variable.
- **Value**. Specifies the value for the particular job variable.
- **Scope**. Indicates the area of the application the job variable relates to, which are classified as job, engine, or system variables.

If you are looking for a job variable of a certain type, you can use the buttons at the top of the Variables tab to filter the contents in the table by the variable's scope. The following options are available:

Icon	Description
J	Toggles the appearance of job variables in the table on or off.
E	Toggles the appearance of engine variables in the table on or off.
S	Toggles the appearance of system variables in the table on or off.

To further facilitate organization of the variables used in your project, the data within the columns of the Variables table can also sorted by clicking any of the column headers to sort the variables in either an ascending or descending ordered sequence.

4.4.4.4 Results tab

The **Results** tab provides a summary of the inputs used and resulting outputs from your process flow, including the XDocs used.

The following buttons are available to help you access your outputs:

Button	Description
	Opens the selected XDoc for viewing in a text editor.
	Opens the selected XDoc for viewing in an XML editor.
	Opens the selected XDoc for viewing in a PDF editor.

Chapter 5

Components

A component is a configurable resource that forms the basis of an OpenText solution. Components can be added to the Development window and linked together in a process flow as required. For more information, see "Process Flows" on page 63.

5.1 Getting Started with Components

Various components can be linked together in a process flow to achieve a result tailored to your specific needs. The topics in this section give more information on some basic concepts about components. For more information on process flows, see "Process Flows" on page 63.

5.1.1 Component Definitions

Component Definition files (or ComponentDefs) are configuration files that store metadata about components to be used in the system. This metadata includes information about how to use, edit, and present a specific component. ComponentDef files end with the `.xComponentDef` extension. Some default ComponentDef files are included with Output Transformation Server. These files are stored in the `<install_home>\install\<version>\initialFiles\system` folder. In Output Transformation Designer, this file system is hidden but is always present so you do not accidentally make changes to core components.

Components that you are able to edit have a corresponding xComponentDef file. For example, the File Event has a corresponding xComponentDef file called `FileEvent.xComponentDef`. This is what `FileEvent.xComponentDef` looks like:

```
 <?xml version="1.0" encoding="UTF-8" ?> <ComponentDef po-
class="com.xenos.framework.system.parm.ComponentDefParm"

 name="FileEvent"

 extension="xFileEvent"

 description="Waits for files to appear and then submits a process flow."

 icon="com/xenos/xserver/gui/editors/resources/directoryListener.png"

 classMain="com.xenos.framework.event.file.FileEvent"

 classParm="com.xenos.framework.event.file.parm.FileEventParm"

 editorDescriptor="com.xenos.xserver.gui.editors.EventListenerEditorDescriptor"

 tabName="Event Listeners/terminalONE"/>
```

This is what the fields mean:

Name	Description
Name	The name for this component type.
Extension	The extension used in the file system to identify this component type.
Description	A description of this type of component.
Icon	A relative path to an image resource located in the file systems. Used to icons for this component in things like tree views. If none is specified, a default icon will be used.
ClassMain	The main Java class for this type of component. This class must implement `com.xenos.framework.component.Icomponent` to function.
ClassParm	The parm object to use for persisting the configuration for this type of component.
EditorDescriptor	The editor descriptor class used to edit this type of component in Output Transformation Designer.
TabName	A path expression used to categorize this component. The path separator is `"r;/"`.

For more information, see "FileEvent" on page 166.

5.1.2 Creating a Component

To create a new component:

1. Creation of new components is handled by the **New Component Wizard**. You can bring up the New Component Wizard within Output Transformation Designer in a number of ways:
   - On the **File Toolbar**, click **New**,  .
   - In the **File Systems** window, right-click a folder in a mounted file system and from the context menu that appears, select **New** > **Component**.
   - Navigate to **File** > **New**.

   The **New Component Wizard** appears.
2. On the **Select a File Type** screen of the New Component Wizard, you must select the component type you want to create. You can use any existing component as a template by selecting it from the file list. The component types are organized by category, but their order can be rearranged by right-clicking **All Components** and from the context menu that appears, selecting your preferred filter type.

   Note: For more information on each component, see "Components" on page 83.

Click **Next**.

3. Some components will offer you the choice of using either **Express Setup** or a **Wizard**. If you are not presented with a choice, refer to the Express Setup section on how to proceed.

   If you selected **Express Setup**:

   a. In the **Name** field, specify a name for your new component. The name should be descriptive of its intended function.
   b. To have the new component placed in a specific subdirectory in the file system, in the **Directory** field either select an existing directory from the drop-down menu, or if you need to create a new directory, enter a new folder name into the field.
   c. When you are complete, click **Finish**.

   If you selected **Use Wizard**, the Wizard contains help dialogs to walk you through the individual steps of the component configuration process. Once you are done, click **Finish**.

The new component is created in the defined directory in the File System window, and is opened in a new tab in the Development window. The component's properties are displayed in the **Development** window, ready to be configured.

5.1.3 Getting Started with Components

Each component has a set of configuration parameters that you can modify to set up and dictate the component's behavior. The parameters for a component are shown in the Properties window, or if you have opened a component on a new tab, in the Development window. Descriptions for each parameter are available directly in the application and are displayed in the Properties window.

Parameter information is stored within the component file, which contains the component type in the file extension to help you and the system quickly identify your components. For example, a Timer Utility component file has `.xTimer` as the file extension.

There are a few ways a component's parameters can be displayed for viewing or editing.

From the **File Systems** window:

- Navigate to the file path location of your component and double-click on the component file. The component's parameters appear on a new tab in the Development window. When you are finished setting the parameter values, remember to **Save** your changes.

From the **Development** window:

- From an open process flow, click on a component. The component's parameters appear in the Properties window. When you are finished setting the parameter values, remember to **Save** your changes.

- Alternatively, from an open process flow, double-click on a component. A dialog box opens showing the component's properties on the **Properties** tab. If available for your particular component, you can switch to the Wizard tab to assist in configuration of the parameters. Click **OK** to close the dialog.

5.1.3.1 Viewing Source Code

You can view the source code for various processes to get a better understanding of how to create custom parameter definitions and runnable components. To view the source code for the test process files:

1. In the **File Systems** window, navigate to the `initialFiles/common/com/xenos/framework/bpengine/test` folder.
2. Double-click on a file to see its source code.
3. If you want to view the component metadata for a particular process, select the associated `.xComponentDef` file.
4. If you want to view the Java code for a particular process, select the associated `.java` file.

   The code appears in a new tab in the Development window.

For more information on writing custom components, see *OpenText Embedded Output Transformation Engine - Developer's Guide (VDTOTS-H-PTE)* and *OpenText Output Transformation Server - Developer's Guide (VDTOTS-H-PGD)*.

Caution

Changing any of the code within a definition file to alter its behavior will impact all sample process flows using it and may result in unwanted results.

5.1.4 Moving and Connecting Components

To move an icon around, click on the component or the middle of the icon and drag it to the desired location.

To draw a connector, click on the small box at the bottom of the icon and drag the mouse towards the desired icon. A straight line appears and changes to a connector arrow once the connection occurs. Arrows should be drawn to connect all the processes in the order you want them to run in.

You can set the zoom percentage with the drop-down menu in the lower left corner of the Development Window.

If you position the pointer over a process icon, a tooltip appears showing the name and type of process it is. Double-click on each process to open the **Properties** to configure the parameters for that component.

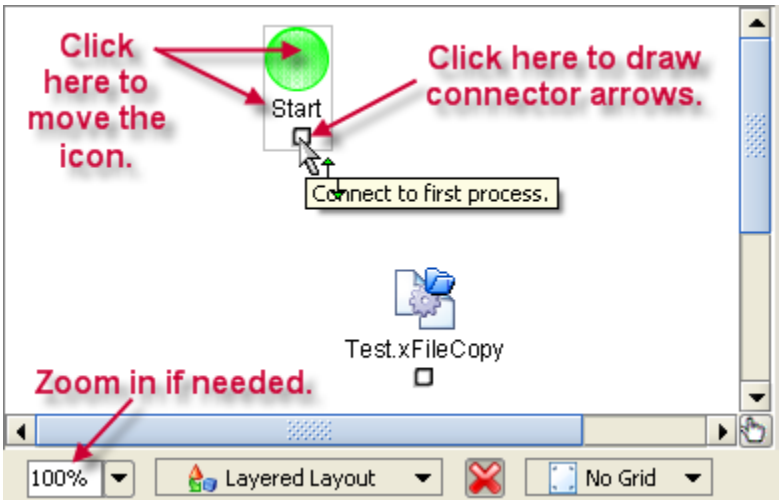


Figure 5-1: *Moving and connecting components example*

5.1.5 Removing a Component Connection

There are three ways to remove a component connection in a process flow:

- Click the connection line, and when the handler appears around it, press **Delete** on your keyboard.
- Right-click the connection line, and when the handler appears around it, select **Remove** from the context menu.
- Remove a component from a process flow. This also removes all connection lines to and from that component. You will have to reconnect the remaining components to make the process flow work.

5.1.6 Annotating Components

Annotations provide information about a component and do not affect the operation of the component in any way. Annotation of components is a best practice especially when multiple partners are working on the same project, or when there are numerous components within a single event or process flow. These notes should provide useful explanations for others on issues such as a component's intended function, or declarations of the component's properties and requirements.

To annotate a component, either hover over a component and select the icon, or right-click on a component and select **Annotate** from the context menu. In the Annotate dialog, enter any notes you want associated with the component, and when complete, click **OK**. To see the annotations for a component, simply hover the cursor over the component and it will appear in the hover text along with the component name and type.

5.1.7 Data Mappings

Configuration of data mappings allows you to map results created by previous processes to the sources needed in the currently selected process.

The **Source and Result Mappings** utility can be accessed by selecting the  icon above a component within a process flow.

All the processes that make up your process flow can be found in the **Processes** drop-down menu in the same order that they will be run.

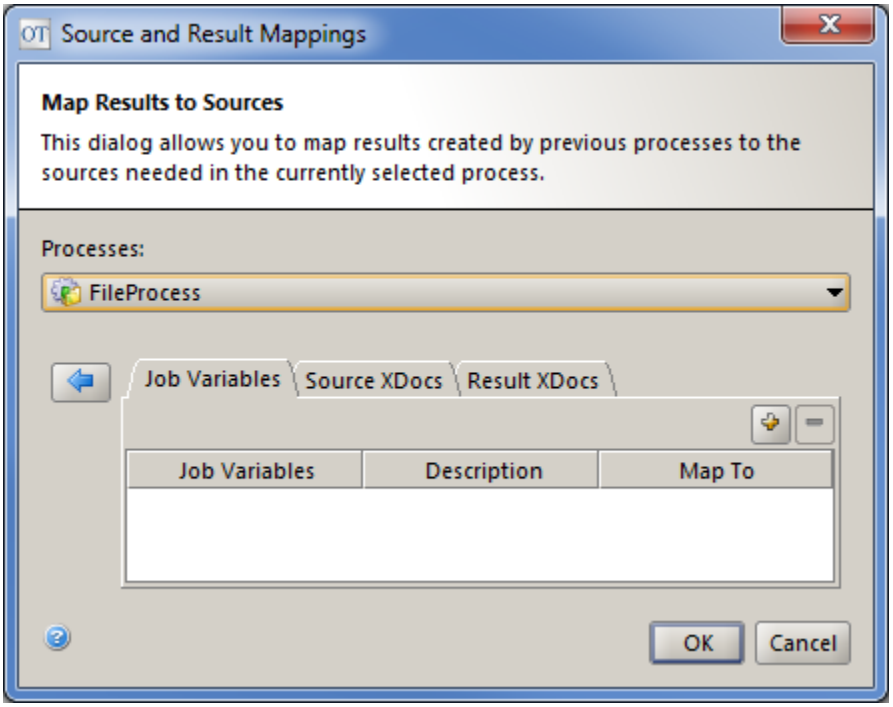


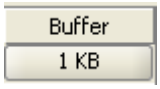
Figure 5-2: Source and Result Mappings dialog

With a process selected, you can configure the process by using the **Job Variables**, **Source XDocs**, and **Result XDocs** tabs to map out its structure.

5.1.7.1 Overriding Default XDoc Buffers

The Source XDocs and Result XDocs tabs include a **Buffer** column, which displays the current XDoc buffer or threshold settings. To override these settings:

1. Click the buffer value to open the **Buffers** tab of the **Mapping Details** dialog.



2. Choose the setting(s) you want to override and enter new value(s) and/or unit(s).
3. Click **OK**.

   The buffer icon will change to indicate there is an override set.

Note: To change the system defaults, set the value(s) in the `default.xSystemConfig` file. For more information, see "System Configuration" on page 17.

5.2 Connectors

Connectors facilitate communication to configured content management systems and provide access to the data and documents stored within these systems. Review the topics in this section to view the types of systems you can connect with using the available connector components.

5.2.1 Repository Adapters

Repository adapters enable communication between Output Transformation Server (or other licensed OpenText products) and third-party repositories for data retrieval from the repository.

5.2.1.1 Standard Repository Adapter Parameters

All repository adapters share a standard set of parameters to control the adapter's basic functionality in relation to memory usage, connection administration, Output Transformation Engine association, and other similar operations. Each repository adapter also has unique configuration parameters that enable connections to be made with the particular server types.

For information on repository adapter-specific parameters, refer to the respective topic pages.

5.2.1.1.1 Standard Repository Adapter Properties

The following standard parameters are available for all repository adapters:

- **AdapterName**. Indicates a unique name identifying this repository adapter.
- **Description**. Indicates a user defined description for the adapter.
- **ResourceHandler**. Details configuration on how resources are cached and whether versioning of AFP resources occurs.
  - **MemoryCacheSizeMeg**. Indicates the maximum memory available, in megabytes, to cache resource groups in memory. A soft reference is used so that the memory is freed if the system requires more memory. A value of 0 means that no memory caching occurs. The default setting is **30**.
  - **DiskCacheSizeMeg**. Indicates the maximum amount of disk space, in megabytes, used to cache resource groups to disk. Cached resources may remain available over a long period of time. A value of 0 means that no disk caching occurs. The default setting is **300**.
  - **DiskCachePath**. Designates the base file path directory to hold cached resources on disk. Repository implementers may create a subdirectory under this location, so all resources are directly tied to specific repositories. The default directory is `./resourceCache`.
  - **DoAfpVersioning**. Indicates whether or not AFP resources versioning occurs on disk. By default, this is **enabled**.
  - **AfpResourcePath**. Designates the base file path directory to hold versioned AFP resources. If this path is concurrently used by another repository, they will share the same AFP Resources Versioner and will not conflict with each other. The default directory is `./afpVersionedResource`.
  - **ResourceAppendString**. Designates the string of text to add to resource file names when writing to disk. If you are writing resources out in different formats, by providing a uniform naming scheme it lowers the risk of your resources becoming desynchronized and conflicting when resources are being written to disk.
  - **AppendMode**. Specifies where the ResourceAppendString value is applied to the resource name. You can choose to add it as either a **PREFIX** (before) or **SUFFIX** (following) to the resource name.
- **AllowAdapterQuery**. Indicates whether or not the adapter and its alias allow queries.
- **ConnectionPool**. Details configuration properties to pool connections for faster response times.
  - **MinConnections**. Indicates the minimum number of connections to keep in the pool while idle. The default setting is **1**.
  - **MaxConnections**. Indicates the maximum number of connections to keep in the pool for future use; this value should not be less than the number for MinConnections. The default setting is **1**.

- **BorrowTimeoutMillis**. Expresses the time, in milliseconds, that a thread will wait for a connection before throwing a timeout exception. If a value of -1 is given, there will be no timeout and the thread will wait indefinitely. The default value is **1000**.
- **EvictionTimeoutMillis**. Expresses the time, in milliseconds, that a connection will remain unused in the pool before the pool is trimmed down to the size indicated in the MinConnections parameter. The default setting is **10000**.
- **EvictionIntervalMillis**. Expresses the time, in milliseconds, between eviction checks. This value should be equal to or greater than the value indicated in the EvictionTimeoutMillis parameter. If a value of -1 is given, there will be no automated eviction check, but eviction will be checked when the pool is used. The default setting is **30000**.
- **OverMaxMode**. Specifies the course of action to take when the maximum number of connections has already been reached and another connection is requested. **BLOCK** will halt the thread until one is available, while **ALLOC_NEW** will allocate a new connection, but will not place it in the available pool after being used.
- **KeepAliveMillis**. Indicates the time interval, in milliseconds, between idle checks for connections in the pool.
- **TestOnBorrow**. Specifies whether every connection is tested before use and creates new ones when necessary.
- **InitialConnection**. Indicates optional user, password, and folder information with which the connection pool is initialized.
    - **User**. Specifies the user ID to use to connect to the database. If a user name is not required to initialize the connection pool, leave this value blank.
    - **Password**. Specifies the password that corresponds to the above User property.
    - **Folder**. Indicates the folder(s) used to initialize the connection pool. The user and password must also be specified. The connection pool will fill until the value in the MaxConnections parameter is reached using the folder in the order specified. If you want an individual name more heavily weighted than the others, repeat the folder name.
- **Alias**. Designates the list of aliases associated with this adapter. Each Aliases entry specifies a different search location. Consider storing data for different applications in different locations within the same object store so defaults can be setup to distinguish between applications.
  Right-click and choose **Add**.
    - **PostProcessToRun**. Specifies the process flow which will run after a method is invoked on the alias.
        - **Process**. Specifies the name of the process flow to run.
        - **InputName**. Specifies the name of the input XDoc that will go into the process flow.

- **ResourceGroupName**. Indicates the name to use when adding the retrieved document resources to the input map. This name is generated automatically and passed into the project, but can be modified to a custom name.
- **VersionedAfpResourceName**. Indicates the name to use when adding the retrieved versioned AFP resources as a job variable.
- **ResultName**. Specifies the name of the output XDoc from the process flow.
- **JobVars**. Defines additional job variables to be added in the job ticket, if required.
- **Token**. Specifies the data type that the user requests for the output. If there will be a choice of output formats required, you must create more than one PostProcessToRun. This can be any supported format, i.e. TIFF, PDF, etc.

5.2.2 AppEnhancer Connectors

The AELoader component facilitates connection to an AppEnhancer environment where you can load documents on to the server.

5.2.2.1 AppEnhancer Loader component prerequisites

To set up AppEnhancer Loader to connect with an AppEnhancer instance, you must have the following additional elements ready:

- An Output Transformation Server license with permissions enabled for the AppEnhancer component
- A fully installed and configured AppEnhancer instance, which must contain the base product as well as the following additional components:
  - AppEnhancer Administrator, AppEnhancer Web Access Server, and AppEnhancer REST Services. (For more information on installing these AppEnhancer products, see *OpenText™ AppEnhancer Installation Guide*.)

Note: For the latest version compatibility information, refer to the Product Compatibility Matrix in the Release Notes.

5.2.2.2 Configuring the AppEnhancer Loader component

The AppEnhancer Loader component must be configured with your server info, user credentials, and information about your load package before you can connect with your AppEnhancer instance.

The following parameters must have values added to them to facilitate the connection and conduct loading to the AppEnhancer server.

- **AuthType**. Specifies the authentication type to use for the REST loader connection.
- **Username**. Specifies the username for the AppEnhancer data source. The method of authenticating your credentials depends on your AppEnhancer data source security configuration.
- **Password**. Specifies the password for the associated user to the AppEnhancer data source. The method of authenticating your credentials depends on your AppEnhancer data source security configuration.
- **HostURL**. Indicates the host URL for the server to connect to. AppEnhancer REST services must be running on the server. If you installed AppEnhancer REST services on the default path, then `http://<hostname>/AppEnhancerREST` is the default host URL. SSL protocols are supported if the JVM and necessary security certificates are properly configured.
- **DatasourceName**. Denotes the name of the data source to load onto the server. When the Username, Password, and HostURL parameters are filled in correctly, clicking the ellipsis button retrieves a list of available data sources to select from.
- **ApplicationName**. Denotes the name of the application to load onto the server. When the Username, Password, and HostURL parameters are filled in correctly, clicking the ellipsis button retrieves a list of available Applications from the data source to select from.
- **IndexFile**. Designates the file path and name of the index file for the load package. Note that this is not the input file itself, however, the index file contains information about the input file. The index file format should be in the Generic index format created by Output Transformation Engine.
- **AuditFileName**. Indicates the full file path and name of the audit file to create. If the file already exists, it will be overwritten. The audit file logs all load operations and can be used for load troubleshooting or load reconciliation.

The following parameters related to document handling options, batch creation, and date formats are optional and do not need to be set if they do not meet your project requirements.

- **DocumentOptions**. Contains options related to document handling during loading.
  - **OnError**. Designates the behavior when an error is encountered while loading documents. The following options are available:

    - **FAIL**. Specifies that the loading process stops. This is the default behavior.
    - **CONTINUE**. Specifies that the loading process continues and any document related failures are skipped. Warning messages are still logged for failures.
  - **OnDuplicateIndexEntry**. Designates the behavior when the document being loaded already exists in the AppEnhancer instance.
    - **DO_NOTHING**. Indicates that AppEnhancer returns an error and does not load the document. During loading, no extra flags are set to handle the duplicate index entries. This is the default behavior.
    - **APPEND_EXISTING**. Indicates that pages with duplicate indexes are appended to the existing document in the target application. During loading, the MergeDuplicateEntry flag is set to handle the duplicate index entries.
    - **NEW_DOCUMENT**. Indicates that a new document with a duplicate index is created in the target application. During loading, the IgnoreDuplicateIndex flag is set to handle the duplicate index entries.
  - **KeyReferenceUpdate**. When enabled, designates that new index values are added to the key reference table if the key value is not found, and data reference fields are updated if the key value is found. If disabled, new index values are added to the key reference table only if the key value is not found. By default, this parameter is disabled.
  - **IgnoreDlsViolation**. Indicates that document level security for the selected application is bypassed and loading is allowed to continue when the user and index information do not satisfy the application's document level security requirements.
  - **COLDFormName**. Designates the name of a form overlay to associate with the batch AELoader upload. The overlay name is also associated with all pages in a document. This value must be a valid overlay name that already exists in the AE_FORMS application on the AppEnhancer server. When the Username, Password, HostURL, and DatasourceName parameters are filled in correctly, clicking the ellipses button retrieves a list of available form overlays from the data source to select from.
  - **SubmitFullText**. Indicates whether documents created from the uploaded batch are sent to the Full Text Queue. By default, this option is turned off.

- **BatchOptions**. Contains options related to batch creation.
  - **BatchIDProperty**. Denotes the name of the AppEnhancer Application field to write the generated Output Transformation Server load ID. You must have a field with this name predefined in the Application before using this parameter. This property is used for auditing and load reconciliation.

    If this parameter is left blank, no load ID is saved with the document index record, which may cause confusion. The source of the uncertainty is due to AppEnhancer being unable to link loaded documents to a given Output

Transformation Server load. However, the reconciliation can also be performed using the AELoader audit log since it records the document IDs that were created for the load process.

- **BatchIDFormat**. Designates the format of the load ID generated by Output Transformation Server. By default, a universally unique identifier is generated by the AELoader and placed in the job variable `AEBatchUuid`. When this value is not set, the default format `OtsLoad: ${AEBatchUuid}` is used.
- **BatchSizeMode**. Indicates how input files (LineData only) are split. You can choose from AUTO or CUSTOM. **AUTO** denotes that when the number of pages in the input file (based on the index file page count) is greater than the AppEnhancer page limit of 32,767 pages, then the file is split into upload batches of approximately equal pages. **CUSTOM** allows you to specify the number of pages to include in an upload batch when used in conjunction with the BatchCustomSize property.
- **BatchCustomSize**. Specifies the maximum number of pages included in an upload batch. When setting this value, you must set BatchSizeMode to CUSTOM otherwise this value is ignored.

- **DateFormat**. Contains options related to the conversion of date fields. When these parameters are used, any fields with the Date data type are converted using the properties from SourceDateFormat and TargetDateFormat. Be aware that when using the date conversion feature, you must set values for both SourceDateFormat and TargetDateFormat. If these parameters are blank, then all index values are passed through as is.
  - **SourceDateFormat**. Designates the format of input date fields. This parameter uses Java SimpleDateFormat syntax, which is also used to interpret date fields.
  - **TargetDateFormat**. Designates the target format for date fields when loading into AppEnhancer. All date index values are converted to this format when documents are loaded.
- **ReliableTransfer**. Contains options related to the reliable transfer mechanism that automatically restarts or resumes a transfer if an error occurs. For more information, see *OpenText Output Transformation Server - User Guide (VDTOTS-H-UGD)*.

5.2.2.3 Reading the AppEnhancer Loader audit logs

The AppEnhancer Loader audit file logs all load operations, which can be useful for troubleshooting or load reconciliation. A single record is added to the log file for each REST call that occurs and a status code representing the last known state of the particular REST call is shown so you can easily determine the current standing.

Records are displayed in the following format:

```
<Date>,<Hash>,<IndexId>,<Code>,<AEdocId>,<BatchId>,<AEdocPageCount>,
<TextInfoField>
```

The record fields are defined as follows:

- **<Date>**. Specifies the current date.
- **<Hash>**. Used to help validate the integrity of a record for the <IndexId>, <Code>, and <AEDocId> fields when handling reload/unload scenarios.
- **<IndexId>**. Specifies the index ID generated by the AELoader to uniquely identify an uploaded document page.
- **<Code>**. Designates the status code. A valid event-related code should always be displayed; NUL is an indication of a code error. See the following descriptions for the meaning of each code:
  - **NUL**. Null. When this is shown, it indicates an error in the record.
  - **UBS**. Upload_Batch_Start.
  - **UBE**. Upload_Batch_End.
  - **CDS**. Create_Document_Start. If KeyReferenceUpdate is disabled and reference data fields are removed from the load request, a dash (-) symbol is used in place of the CDS record to indicate that it has been omitted.
  - **CDE**. Create_Document_End. If KeyReferenceUpdate is disabled and reference data fields are removed from the load request, a dash (-) symbol is used in place of the CDE record to indicate that it has been omitted.
  - **KRU**. Key_Reference_Update. If KeyReferenceUpdate is enabled and existing key and reference data is found in an uploaded document's index information, this code is appended to the new data reference values.
  - **CDEM**. Create_Document_End_Merged. Indicates a loaded document was merged into an existing document. In the audit records, the document ID represents the ID of the merged document.
  - **STR**. Started. Denotes an AELoader process was started.
  - **ABR**. Aborted. Indicates that the AELoader was stopped due to a user manually aborting an AELoader process.
  - **ERR**. Errored. Designates that the AELoader was stopped due to an error occurring during loading.
  - **FIN**. Finished. Indicates that the AELoader stopped and finished loading with no errors.
  - **SKIP**. Skipped. Denotes a document was skipped and not loaded.
  - **TCON**. Test_Connection. Denotes when a connection error occurs.
  - **IVAL**. Index_Validation. Designates when an error occurs during AELoader's index validation process.
  - **DVAL**. Document_Load_Validation. Designates when an error occurs during AELoader's document load validation before a load retry attempt.
  - **INFO**. Displays informational messages, such as Index Field information.

  - **IDS**. Index_Document_Start. Indicates the starting point of an index document.
  - **IDE**. Index_Document_End. Indicates the ending point of an index document.
  - **CBS**. Create batch start. Only used when processing Linedata.
  - **CBE**. Create batch end. Only used when processing Linedata. The page count and temporary batch file location are also displayed.
- **<AEdocId>**. Denotes the doc ID assigned by AppEnhancer to the particular document. If an ID is not assigned, -1 is used.
- **<BatchId>**. Denotes the batch ID assigned by AppEnhancer to the uploaded batch file.
- **<AEdocPageCount>**. Designates the current page count of the document after a successful call.
- **<TextInfoField>**. Specifies any general information such as index fields or error messages. The field is not always populated, but events that result in a CDE status code display index information while NUL events display information related to the error.
-

5.2.2.4 Setting up the AELoader application browser

For use with AELoader, an application browser is available to view the data source's applications and properties so users can easily verify aspects like property names or field definitions for their indexes. A connection to AppEnhancer must be established through the AELoader component before you can use the browser.

To access the browser:

1. Set up the **AELoader** component as usual. (For more information on setting up the component, see "AppEnhancer Connectors" on page 92.)
2. When configuring the AELoader parameters, pay special attention to the **DatasourceName** parameter since this indicates the name of the data source to access and an AppEnhancer server may have multiple data sources.

   You can enter the DatasourceName value manually, or alternatively, once you have configured your server details and user credentials and the component can access the server, you can click the DatasourceName parameter's ellipsis button to bring up the **Property – DatasourceName** dialog box, which displays a list of all available data sources on the server. Select a data source and click **OK**.
3. **Save** your changes to AELoader.
4. At the bottom of the Development window, switch to the **Browser** tab.
5. On the Browser tab, click **Load**.

   A list of Applications is retrieved from your AppEnhancer instance and is shown in the browser's Applications pane.

5.2.2.5 Using the AELoader application browser

The AELoader application browser displays a list of the Applications stored in your AppEnhancer instance's data source, as well as their associated index fields and field definitions. The browser allows users to achieve a number of tasks within the same interface, such as browsing index field names for verification or viewing date layouts to determine the proper mapping or conversion format for them.

Once you have established a connection to the server for your browser and you have loaded the Applications from your server, you can click on any item in the **Applications** list and then its associated index fields and related properties appear in the right pane. The following properties are shown for each index field:

- **Field Name**. Specifies the name of the index field.
- **ID**. Denotes the internal field ID for the application.
- **Type**. Specifies the index field type.
- **Length**. Specifies the maximum number of alphanumeric characters that an index field may contain.
- **Mask**. Indicates any type of validation masking to specify the character formatting for the associated field. For example, a Date type field can have a mask to display all dates in a consistent format like `MMMM DD, YYYY`.
- **Required**. Indicates whether the field is mandatory for the index. Indexes can be built or loaded at any time if they do not contain any mandatory fields.

You can also sort the index fields by their **Field Name**, **ID**, **Type**, **Length**, or **Mask** values by clicking on the column header.

The following tasks can also be performed in the AELoader application browser:

- **Load**. Retrieves the Applications in the specified data source from the AppEnhancer server.
- **Use**. Copies the name of the selected Application to AELoader's ApplicationName parameter.

5.2.3 Content Management Interoperability Services Connectors

The **CMISAdapter** component allows you to control and interact with files and documents stored on document management repositories that adhere to the Content Management Interoperability Services (CMIS) standards protocol. The CMISAdapter supports the following types of CMIS servers (required versions listed as needed):

- Alfresco 3.3+
- Microsoft SharePoint Server 2010

5.2.3.1 Setting up the CMISAdapter to Connect to a Repository

Using the CMISAdapter to connect to any of the supported repositories through the Archive Navigator does not require any external configuration. All set up and management of the repositories can be performed within Output Transformation Designer.

Note: Your CMIS application server must already be up and running before you can connect to it with the CMISAdapter.

To connect to a repository using the CMISAdapter component:

1. In the **File Systems** window, navigate to your instance of the **CMISAdapter** component.

   Note: If you do not have an existing instance, you must create a new one. For more information on performing this task, see "Creating a Component" on page 84.

   A new tab showing the CMISAdapter's parameters displays in the **Development** window.

2. The component's parameters control the behavior of your component as well as contain configuration settings to assist in the set up process. The basic parameters that need to be configured in order to set up a connection to the database are as follows:

   - **ATMOPubUrl**. Indicates the URL or connection string used to connect to your repository.
   - **ConnectionProperty** > **Name**. Indicates the name of the connection property.
   - **ConnectionProperty** > **Value**. Indicates the value of the connection property.

3. Depending on the requirements for your project, you may wish to configure some of the other parameters. For more information on the parameters and their functions, see CMISAdapter Properties.

   Once you have finished configuring your settings, click **Save**.

4. Next, you must start the adapter by selecting the **Execute** button,  , on the toolbar.

   The **Execution Options** screen appears.

5. Click **Execute**.

   An output tab for the component appears in the Events window displaying status messages about the startup process. At the end of the procedure, a message stating **Finished initializing Repository Service Adapter** appears.

5.2.3.2 CMISAdapter

The CMISAdapter component allows you to control and interact with files and documents stored on document management repositories that adhere to the Content Management Interoperability Services (CMIS) standards protocol.

Note: : When conducting searches for documents through the CMISAdapter, documents within subfolders and any custom properties you have defined are included in the search, but note that you cannot search across multiple document types during a single search request.

5.2.3.2.1 CMISAdapter Properties

- **ResourceHandler**. Details configuration on how resources are cached and whether versioning of AFP resources occurs.
  - **ResourceHandlerType**. Specifies which type of resource handler to use.
  - **Enabled**. Indicates whether a resource handler is used. By default, this is enabled.
  - **DocumentExtension**. Specifies the file type extension of the retrieved document.
  - **DiskCachePath**. Designates the base file path directory to hold cached resources on disk. Repository implementers may create a subdirectory under this location, so all resources are directly tied to specific repositories. The default directory is `./resourceCache`.
  - **ResourceAppendString**. Designates the string of text to add to resource file names when writing to disk. If you are writing resources out in different formats, by providing a uniform naming scheme it lowers the risk of your resources becoming desynchronized and conflicting when resources are being written to disk.
  - **AppendMode**. Specifies where the ResourceAppendString value is applied to the resource name. You can choose to add it as either a **PREFIX** (before) or **SUFFIX** (following) to the resource name.
- **ConnectionPool**. Details configuration properties to pool connections for faster response times.
  - **InitialConnection**. Indicates optional user, password, and folder information with which the connection pool is initialized.
    - **User**. Specifies the ID to use to connect to the database. If a user name is not required to initialize the connection pool, leave this value blank.
    - **Password**. Specifies the password that corresponds to the above User property.
    - **Folder**. Indicates the folder(s) used to initialize the connection pool. The user and password must also be specified. The connection pool will fill until the value in the MaxConnections parameter is reached using the

folder in the order specified. If you want an individual name more heavily weighted than the others, repeat the folder name.

- **AllowAdapterQuery**. Determines whether or not the adapter and its alias can be queried. By default, this is enabled.
- **PostProcesses**. Contains configuration settings for the ViewAs feature.
  - **AllPostProcesses**. Contains configuration settings for all the logically named post processes defined in the repository instance.
    - **LogicalName**. Designates the name of the Output Transformation Server runnable to handle the conversion.
    - **EsRunnable**. Indicates the name of the process flow or runnable to execute that will handle the conversion.
    - **InputXDocName**. Specifies the name of the input XDoc to use for your runnable.
    - **ResourceXDocName**. Denotes the name of the resource XDoc that the runnable is expecting.
    - **VersionedAfpResourceName**. Indicates the name to use when adding retrieved versioned AFP resources as a job variable.
    - **OutputXDocName**. Specifies the name of the output XDoc to use for your runnable.
    - **ReturnType**. Designates the file type the resulting output should be.
    - **JobVars**. Specifies the job variables that are used by the post process.
      - **Name**. Indicates the name of the job variable.
      - **Value**. Indicates the value of the job variable.
  - **ConversionsFrom**. Contains settings related to file type conversions.
    - **ExtFrom**. Denotes the file type you wish to convert from.
    - **DefaultTo**. Identifies the default method used to view this type of file. A blank value means that no conversion is performed.
    - **ConversionsTo**. Contains settings related to all possible output formats.
      - **ExtTo**. Denotes the file type you wish to convert to.
      - **OrderedList**. Identifies the logically named post processes (typically with variables) in an ordered list, which are tested sequentially. If a match is found, the conversion is performed.
- **Authentication**. Contains options relating to user authentication.
  - **UseEsAuthentication**. Designates whether or not the Output Transformation Server authentication engine should be used for initial authentication of users.

- **ATMOPubUrl**. Indicates the URL or connection string used to connect to your repository.
- **ConnectionProperty**. Contains configuration options relating to CMIS connection properties.
  - **Name**. Indicates the name of the connection property.
  - **Value**. Indicates the value of the connection property.

Note: For information on the standard repository adapter parameters, see "Standard Repository Adapter Parameters" on page 89.

5.2.4 Documentum Connectors

The topics in this section discuss working with Documentum connectors.

5.2.4.1 Configuring Output Transformation Server to Connect to a Documentum Server

Connections from Output Transformation Server to a Documentum server require preparations both in and out of Output Transformation Server. The following steps summarize the process:

1. Ensure that the Documentum service is installed and running on your server.
2. Copy the JAR files.
3. Configure the Documentum Foundation Classes.
4. Configure the DocumentumAdapter.

5.2.4.1.1 Step 1: Ensuring that the Documentum service is installed and running on your server

Check that all the necessary portions of the Documentum suite are installed and properly configured on your application server. In addition to the several interconnected elements to the installation and configuration process for the different products within the suite that need to be completed, you should also install the latest fixes and patches. If you require instructions on these procedures or further information on the most recent patches, refer to your Documentum installation materials or the vendor's website.

5.2.4.1.2 Step 2: Copying the JAR files

The Documentum service utilizes four JAR files that allows communication between our application to the Documentum server. These files can be found in your Documentum installation directory:

- `aspectjrt.jar`
- `certjFIPPS.jar`
- `dfc.jar`
- `jsafeFIPS.jar`

Copy these four JAR files and place them in your `<install_home>\lib\ext` folder.

5.2.4.1.3 Step 3: Configuring the Documentum Foundation Classes

As part of the installation process of the Documentum suite on the server, the installer creates a `dfc.properties` file, which is a Documentum Foundation Classes (DFC) file where further connection properties are stored. The file is typically stored in the config folder of your Documentum installation directory.

Open the `dfc.properties` file for editing, and provide the values for the following properties:

- `domain[host]`
- `dfc.docbroker.port[0]`
- `globalregistry.username`
- `globalregistry.password`

The values for these properties must match the attributes of your application server in order for connections between the servers to be effectual.

5.2.4.1.4 Step 4: Configuring the DocumentumAdapter component

In terms of the DocumentumAdapter, configuration of many of the parameters is ultimately determined by your particular needs; however, there are a few specific parameters that you must set up for Output Transformation Server to successfully connect to the Documentum server.

To configure the DocumentumAdapter to connect with the Documentum server:

1. In order to access the DocumentumAdapter, you must create a new component. Click **File** > **New**.
2. Locate the repository adapter template file type by navigating through to **Connectors** > **DocumentumAdapter**. Select the **DocumentumAdapter** and then click **Next** to open the **Name and Location of New DocumentumAdapter** dialog.
3. In the **Name** field, enter a name for your new repository adapter. Use the **File System** and **Directory** fields to specify where you would like to save your new

repository adapter. When you are done, click **Finish**. Your new repository adapter's parameters appear in the development window.

4. Provide values for the following parameters:

   - **ConnectionFactory**.
     - **DfcPropertiesPath**. Specifies the file path location of the Documentum `dfc.properties` file.
     - **RepositoryName**. Indicates the Documentum repository name.

       Note: This is not the domain or database name.

     - **DefaultFolder**. Specifies the Documentum default folder. A value is not required, but if a folder name is provided, then you must be aware that any searches run will only search within the same path; subfolders are not included in the search.
     - **HitListBack**. Specifies the method of returning properties in the HitList. **Custom** returns properties only for the designated object type, while **All** returns everything.
   - **ConnectionPool > InitialConnection**.
     - **User**. Specifies the user ID to use to connect to the Documentum server. If a user name is not required to initialize the connection pool, leave this value blank.
     - **Password**. Specifies the password that corresponds to the above User property.

5. Configure the other parameters as necessary for your project and when you are finished, **Save** your settings.

Note: For more information on all of the DocumentumAdapter component's parameters, see "DocumentumAdapter" on page 104.

5.2.4.2 DocumentumAdapter

The **DocumentumAdapter** component enables access to data stored on a Documentum server. The DocumentumAdapter is the second-generation version (V2) of the DocumentumRepositoryAdapter.

5.2.4.2.1 DocumentumAdapter Properties

This component includes standard repository adapter parameters. For more information, see "Standard Repository Adapter Parameters" on page 89.

The following parameters are set for the DocumentumAdapter component:

- **PostProcesses**. Configuration for ViewAs settings.
  - **AllPostProcesses**. Contains configuration settings for all the logically named post processes defined in the repository instance.
    - **LogicalName**. Designates the name of the Output Transformation Server runnable to handle the conversion.
    - **EsRunnable**. Indicates the name of the process flow or runnable to execute that will handle the conversion.
    - **InputXDocName**. Specifies the name of the input XDoc to use for your runnable.
    - **ResourceXDocName**. Denotes the name of the resource XDoc that the runnable is expecting.
    - **VersionedAfpResourceName**. Indicates the name to use when adding retrieved versioned AFP resources as a job variable.
    - **OutputXDocName**. Specifies the name of the output XDoc to use for your runnable.
    - **ReturnType**. Designates the file type the resulting output should be.
    - **JobVars**. Defines additional job variables to be used in the post process.
      - **Name**. Indicates the name of the job variable.
      - **Value**. Indicates the value of the job variable.
  - **ConversionsFrom**. Contains settings related to file type conversions.
    - **ExtFrom**. Denotes the file type you wish to convert from.
    - **DefaultTo**. Identifies the default method used to view this type of file. A blank value means that no conversion is performed.
    - **ConversionsTo**. Contains settings related to all possible output formats.
    - **ExtTo**. Denotes the file type you wish to convert to.
    - **OrderedList**. Identifies the logically named post processes (typically with variables) in an ordered list, which are tested sequentially. If a match is found, the conversion is performed.
- **Authentication**. Contains options relating to user authentication.
  - **UseEsAuthentication**. Designates whether or not the Output Transformation Server authentication engine should be used for initial authentication of users.

- **ConnectionFactory**. Sets the connection parameters required to connect to your existing Documentum server.
  - **DfcPropertiesPath**. Specifies the file path location of the Documentum `dfc.properties` file.
  - **RepositoryName**. Indicates the Documentum repository name. Note that this is **not** the domain or database name.
  - **DefaultFolder**. Specifies the Documentum default folder. A value is not required, but if a folder name is provided, then you must be aware that any searches run will only search within the same path. Subfolders are not included in the search.
  - **HitListBack**. Specifies the method of returning properties in the hit list. **Custom** will return properties only for the designated object type, while **All** returns everything.

5.2.5 Third Party Connectors - IBM

The topics in this section describe the connectors available to help you to connect with IBM's products.

5.2.5.1 FileNet Connectors

Topics included in this section describe the FileNet connectors.

5.2.5.1.1 Preconfiguration for FileNet components

Before using the **FileNetP8Loader** component to upload files to FileNet P8, some additional configuration must be completed. The procedure depends on the application server you are using. Follow the relevant procedure.

5.2.5.1.1.1 Standalone configuration with Output Transformation Designer

To configure Output Transformation Designer for use with the FileNet following installation of Output Transformation Server, do the following:

1. Copy the required P8 JAR (Java Archive) files IBM FileNet supplied into the appropriate `ext` folder.

   The default path on Windows is the `<install_home>\lib\ext` folder.

   The default path on Unix is the `<install_home>/lib/ext` folder.

   We do not supply these JAR files as they are covered under IBM license. The JAR files are located in your FileNet installation directory. The following is a list of all the files that must be copied:

   - `javaapi.jar`
   - `jace.jar`
   - `jaxrpc.jar`(or `jaxpc_p8_wsi.jar`)
   - `jetty.jar` (or `jetty_p8.jar_wsi.jar`)

- saaj.jar (or saaj_p8_wsi.jar)
- wasp.jar
- wsdl_api.jar

javaapi.jar and jace.jar can be found in your FileNet P8 Workplace application on the source system. The other JAR files can be found in the wsi folder under the Content Engine on the source system.

Note: When copying these jar files, precede the filename with 'FileNet_' to avoid conflicts with other filenames. For example, the file wasp.jar in the FileNet installation directory will be named FileNet_wasp.jar in the Output Transformation Server install directory.

2. Edit the \lib\auth.conf file.

   The default location for the auth.conf file is the <install_home>\lib\ folder.

3. Open the file in a text editor.

4. Add the following lines:

```
FileNetP8 {
com.filenet.api.util.WSILoginModule required;;
FileNetP8WSI {
com.filenet.api.util.WSILoginModule required;
};
```

Note: The logicalName field needs to correspond to one of these values: "FileNetP8" or "FileNetP8WSI".

5.2.5.1.2 FileNetP8Loader

The **FileNetP8Loader** is a component used to upload documents into IBM FileNet P8 (FileNet P8).

The FileNetP8Loader can load any object into an Enterprise Content Management (ECM) system, including some of the more common document types such as PDF, AFP, and METACODE. The FileNetP8Loader loads the documents and any metadata that is defined in the index file for the documents.

The FileNetP8Loader loads individual objects into FileNet P8. It can load individual source objects or can split stacked source object files into individual objects during loading. Source objects can be loaded directly or printstreams can be converted ahead of time from any supported input format to any supported output format using Output Transformation Engine.

Tip: By default, a FileNetP8Loader component will create a folder specified in the FileNetP8Loader configuration/index file on a FileNet ObjectStore if the folder does not exist already. Since FileNetP8Loaders use batch updates to upload documents, the folder will not be committed until a batch is committed. When running concurrent FileNetP8Loader components, create the folder needed to avoid any conflicts.

The documents that the FileNetP8Loader loads can be retrieved using the "FileNetP8Adapter" on page 133 or by other applications. Neither the FileNetP8Adapter, nor the FileNetP8Loader transforms documents. You will need an Output Transformation Project or a custom process within a process flow to transform the documents using OpenText products.

5.2.5.1.2.1 FileNetP8Loader requirements

The following are FileNetP8Loader requirements:

Supported Versions and Platforms	For the most up-to-date listing of compatible IBM FileNet P8 versions and platforms, refer to the Release Notes.
Supported Application Servers	The FileNetP8Loader uploads files from Output Transformation Designer or through a Output Transformation Server JEE application deployed to an application server. For more information on the supported application servers, see the Release Notes.
Output Transformation Server Licensing Requirements	A Output Transformation Server build with the FileNetP8Loader component must be installed and licensed. To ensure that your Output Transformation Server license has the FileNetP8Loader enabled: 1. Navigate to **Help** > **About** > **License** in Output Transformation Designer. 2. Expand the **Output Transformation Server** section in the left pane. 3. Ensure **ESP8LD** is in the list of components.

For more information about OpenText license keys, see *OpenText Output Transformation Server Installation Guide*.

5.2.5.1.2.2 Preparing FileNet P8 for document loading

For FileNet P8 to receive documents uploaded using the FileNetP8Loader component, there must be a document class defined in FileNet P8 for the upload. A document class is a group of properties that help describe a document in FileNet. It can be extended to allow for additional properties and it derives all the properties of its parent document class.

The properties that need to be set up are the index data, or the document metadata. For example, an insurance claim form may have index data such as **ClaimantName**, **Date**, and **Amount**. This data would be stored separately so that a FileNet P8 user can do a search and find the related document.

Each document class may include default properties, such as the DefaultDocumentClass, that describe FileNet, with properties that define the class unless overwritten by index data.

Note: The DefaultDocumentClass that you create in FileNet P8 should have attributes that match the index key/value pairs in the index file being used. These key value pairs will be stored in the attributes by the same name to facilitate searching. The index keys are the **symbolic names** as defined in the FileNet P8 system.

If you want to load resources, create a second document class to handle resources only. For more information, see "Configuring a Document Class for Resources Using FileNetP8Loader" on page 110.

5.2.5.1.2.3 Creating and Configuring a Document Class

To create a new document class:

1. In the **Object Stores** folder on the left sidebar, right-click the **Document Class** folder and click **New Class** to open the **Create a Class Wizard**.
2. In the **Name** field, enter a name for your document class.

   Notice that the **symbolic name** is automatically given the same name, with any spaces removed. Remember the **Symbolic Name** as you will need to enter it in the **DefaultDocumentClass** field of the FileNetP8Loader component properties
3. Optionally, you may also provide a brief summary of the new class in the **Description** field.
4. Click **Next** to open the **Select Properties** dialog.
5. All of the document properties related to the current document class are displayed. You can choose from existing properties, or create a new one specific to the document class's requirements.

   Note: All properties have an associated data type (for example, String, Integer, and Date). Take care when selecting an existing property as changes will affect properties already in use.

   If you want to add an **existing** property, from the **Available** group menu select the property and click **Add** to add the selected property to the document class.

   If you want to add a **new** property:

   a. Click **New** to open the **Create a Property Template Wizard**.
   b. Enter a name for the new property in the **Name** field.

      The **Symbolic Name** is automatically given the same name as the one provided in the Name field, with the exception of spaces. It is important to make note of this Symbolic Name as well, since it must be entered in your FileNet P8 Loader index file in order to upload the metadata.
   c. Click **Next** to open the **Select the Data Type** dialog.
   d. From the list, select the data type that defines the kind of data the property template will contain. After the property template is created, the data type cannot be changed.
   e. Click **Next**.
   f. Choose whether the property will be a **Single Value** or a **List**. In most cases, a single value will suffice.
   g. Click **Finish**.

The new property is created, and is automatically added to the list of properties for this document class. Repeat this procedure to add additional required properties.

6. Click **Next** to open the **Select Property Attributes** dialog.
7. In most situations, the default settings on the **Select Property Attributes** screen can be left as is.
8. Click **Next** to open the **Specify Content Storage Parameters** dialog.
9. The default settings on the **Specify Content Storage Parameters** screen meet most users' requirements and can be left as is.
10. Click **Next**.
11. Configure the behavior of audit logs on the following screen.
12. Click **Finish**.

    The new document class is ready to be used for uploading files.

If the FileNetP8Loader configuration needs a separate document class for resources, define the properties for the document class in the **ResourceGroupOptions** section of the FileNetP8Loader configuration. This section contains information the FileNetP8Loader will use when dealing with loads that require resources. The parameters specified within this section will create the document class for the resources.

Note: For users who are performing a migration to FileNet, it is recommended to give the same property names as those found in the original source.

5.2.5.1.2.4 Configuring a Document Class for Resources Using FileNetP8Loader

To configure a new document class for resources within the FileNetP8Loader configuration, enable the **ResourceGroupOptions** section with properties for the document class and resource bundle. For more information, see "FileNetP8Loader component properties" on page 111.

ResourceGroupOptions	Identifies the grouping of properties needed when resources are being stored. In order to minimize the amount of storage space that is used when storing a large number of documents resources can be grouped into a bundle for the entire load.
Enabled	Specifies whether or not the resource grouping feature is to be used for the load.
DefaultResourceGroupDocumentClass	This document class is specifically used for resources within the FileNetP8Loader configuration.
DocumentClass	Specifies the name of the FileNet P8 document class to use for the resource bundle.

PropertyName	Identifies the name of the property within the document class used to store a unique reference key.
DefaultResourceGroupKey	Specifies the key to be used to uniquely identify the resource bundle. If you are using Output Transformation Engine in your process flow, you need to create a key to be used by the FileNetP8Loader when loading resources and enter that key here, in the form ${ResourceKey}. ES comes with a built-in IOHandler called **com.xenos.d2e.fwserver.store.ResourceDigestIOHandlerFactory** which is an implementation of **UserIoHandlerFactory**. Use this handler to create the resource key.
DefaultResourceGroupPath	Specifies the location of the resource bundle on the local files system.
DefaultResourceGroupFolder	Indicates the folder that will be created to store the resources in FileNet.
DocTypeProperty	Specifies a logical grouping of properties used to specify a Document Type.
DocTypePropertyName	Identifies the name of the property in the document class to be used to hold a Document Type's information.
DocTypePropertyValue	Specifies an actual Document Type, such as AFP, PDF, or PDFx.
ResourceBundleProperty	Indicates the name of the FileNet property in the Document Class to be used to store a reference to the resource bundle that was loaded.

5.2.5.1.2.5 FileNetP8Loader component properties

The FileNetP8Loader properties define the loader's options in your process flow. These properties mirror the properties in an index file, which defines metadata for document objects, such as PDF, AFP, or Metacode batch files, the FileNetP8Loader will load. The properties for both the index file and the FileNet P8 document class must match for loading of object files to happen.

The FileNetP8Loader has the following parameters:

- **FNetAdaptorFactory**. Indicates the configuration to connect to the FileNetInstance using filenet.wcm API.
  - **LogicalName**. Specifies the unique name of the FileNet Repository Adapter. Adapters are specified within the `auth.conf` file in the `/lib/ folder` in your Output Transformation Server install directory.
  - **UriConnectionString**. Specifies the connection string to your FileNet server. If you are connecting a FileNet P8 3.5.2, this value may be null.
  - **ApiConfigFile**. Specifies the API configuration file that tells the FileNet Repository Adapter how to connect to FileNet P8 3.5.2. The file required can be found in your IBM FileNet install directory under the file name `WcmApiConfig.properties`. If you are connecting to FileNet P8 4.0.2, this value may be null.

  - **DomainName**. Indicates the domain name of the current FileNet P8 instance.
  - **Userid**. Specifies the Userid to use to connect to the FileNet server.
  - **Password**. Specifies the password that corresponds to the Userid property.

- **RootFolder**. Indicates the base folder from which all documents are loaded.
- **DefaultDocumentClass**. Specifies the symbolic names of the FileNet P8 document class where files will be uploaded.

  Note: A value entered in the IndexFile parameter takes priority over this parameter, and overrides this value.

- **DefaultObjectStore**. Specifies the object store that contains the document class to load all uploaded files to.

  Note: A value entered in the IndexFile parameter takes priority over this parameter, and overrides this value.

- **DefaultMimeType**. Specifies the mimetype of the documents to upload.

  Note: A value entered in the IndexFile parameter takes priority over this parameter, and overrides this value.

- **DefaultFolder**. Specifies the FileNet P8 folder to upload all documents to, unless overridden by the index file. If you have designated RootFolder, this default folder will be contained within the root folder that you have specified.

  Note: A value entered in the IndexFile parameter takes priority over this parameter, and overrides this value.

- **IndexFile**. Specifies the index file containing all the documents to load.
- **IsIndexFormatIbmOnDemand**. Indicates the index file has an IBM OD format (compatibility mode).
- **RemoveSpaces**. Indicates whether spaces in object store, document class, and property names are removed. This mimics the FileNet relationship between the Display name and Symbolic name. For example, a document class may appear as having the name “Doc Class”, but within the system it is written as “DocClass”.
- **LoadUnfiled**. Indicates whether the FileNetP8Loader component ignores folder options specified at **RootFolder**, **DefaultFolder**, or in any index records. Documents will be loaded into FileNet without a folder relationship. Users can see the documents from the Unfiled tree node within FileNet Enterprise manager and can search for the documents by their FileNet property values. This setting is disabled by default.
- **PageQueries**. Indicates whether or not FileNetP8Loader will use the FileNet paged query feature.
  When set to **false** paging will not be used and the number of results returned by a query will depend on the `ServerCacheConfiguration.NonPagedQueryMaxSize` property on the FileNet Server. If a high volume of records are being loaded, this option may be memory intensive.

When set to **true** the query will be paged and will return all matching objects. The number of objects returned to FileNetP8Loader per page will depend on the `ServerCacheConfiguration.QueryPageDefaultSize` property on the FileNet Server. Using paged queries allows P8 to retrieve data in intervals until the entire result set is returned. This may reduce the memory requirements on the server.

- **DefaultDateFormat**. Indicates the default date format to use for parsing dates in an index file when the Date property is defined within the FileNet server. When the date is stored as a string, use of this parameter is not required. Different types of systems may use mismatched date formats when creating index files, therefore, this parameter provides the ability to parse dates based on a variety of formats. It is highly recommended that you enter a default format value since if a format is not specified or it parses unsuccessfully, the FileNetP8Loader will systematically attempt to use the date formats listed in the mapping table in sequential order. If the date cannot be parsed, an exception will be thrown.

  Note: MMMM returns the full name of a month. For instance, MMMM dd, yyyy would appear as February 6, 2008.

  You can choose a default date format from the provided list, or write and implement a custom date parser. The custom parser will implement the IDateParser interface. Enter the path and classname into the DefaultDateFormat property. If there are multiple date formats in the index file or the date format does not exactly match what the custom parser is expecting you will encounter an error.
- **BatchIDProperty**. Specifies the name of the property in the document class used to store a batch/load ID.
- **BatchIDSuffix**. Specifies a suffix to append to append to the BatchID. This is useful for making a distinction between BatchIDs when the loader is running concurrently on multiple non-Output Transformation Server-clustered instances.
- **ContinueOption**. Contains settings related to continue on error options.
  - **OnErrorContinue**. Indicates when enabled that the FileNetP8Loader will continue to load documents even when any of the following errors occur:
    - The object store in the index or configuration file does not exist on the P8 server.
    - The document class specified in the index or configuration file does not exist in the object store specified in the index or configuration file.
    - A property specified in the index file does not exist in the specified document class.
    - A property (property template) has a choice list associated with it and the value provided in the index for that property does not exist in the choice list.

    Errors will be written to the log file specified below.

    Do not use with Reliable Transfer> 0 retries as it will cause exception to be thrown.

- **FailedLogFilename**. Specifies the log file to use when OnErrorContinue is selected. The log file contains the following fields:
    - docIndex: number of documents within the index.
    - filename: filename set in the index for the document.
    - objectstore: name of the object store the document is loaded into.
    - foldername: name of the default folder the document is loaded into.
    - documentclass: name of the document class being used to store index information for the document.
    - properties: a list of properties and property values from the index for the document.
    - sourceFilename: path of the source file.
    - sourceLength: number of bytes within the source.
    - sourceOffset: offset within the source for this document's data.
    - errorMsg: a description of the error encountered.
    - stackTrace: the stack trace from the exception that was thrown if one was thrown.

- **ResourceGroupOptions**. Specifies the ResourceGroup configuration.
- **AuditFileOption**. Specifies the path and name of the audit file to write to. The audit file shows the status of each document loaded in the batch, as well as the DataStore loaded to the document's GUID.
    - **FileName**. Specifies the name of the trace file to be written.
    - **OverWrite**. Identifies whether or not to overwrite the existing trace file with the same name specified by FileName. If disabled, each load will append the audit information to the same file separated by hashmarks, #####.
    - **AppendUniqueId**. Identifies whether or not to append a unique id to the end of the trace file name. As long as this is checked, a file with a unique name will be created and written to.
    - **AuditOption**. Contains settings related to the return of various document ID element types that can be written to an XDoc for confirmation or reference purposes after a document is successfully loaded. You can select which document ID types to include in the XDoc.
        - **WriteToXdoc**. Determines whether XML elements should be written to an XDoc in the ResultMap with the default name XDocLogger after a document is successfully loaded. By default, this is **false**.
        - **BatchID**. Indicates whether the BatchID is included in the XDoc. By default, this is **true**.
        - **DocumentName**. Indicates whether the DocumentName is included in the XDoc. By default, this is **true**.

- **DocumentID**. Indicates whether the DocumentID is included in the XDoc. By default, this is **true**.
- **FolderName**. Indicates whether the FolderName is included in the XDoc. By default, this is **true**.
- **ClassName**. Indicates whether the ClassName is included in the XDoc. By default, this is **true**.
- **ObjectStore**. Indicates whether the ObjectStore is included in the XDoc. By default, this is **true**.

- **SummaryFileOption**. Specifies the path and name of the summary file to write to. The summary file contains condensed information about the load, similar to the information shown below:

  OpenText Filenet P8 Loader Summary

  ==========

  RootFolder = /ES_QA_TEST

  StartTime = 2010-03-22 11:35:11

  StopTime = 2010-03-22 11:35:15

  TotalTime = 0:0:4

  FilesPerMinute = 3

  BytesPerMinute = 480409

  Folders Created

  ==========

  /ES_TEST/FN_P8Loader_006

  ObjectStore FolderName DocumentClass MimeType FilesUploaded TotalBytes

  ========== ========== ========== ======== ========== =======

  ObectStore1 FN_P8Loader_006 ES_test application/pdf 3 480.409kb

  Note: The total bytes transferred will always reflect the uncompressed file size even if pre- and/or post-transfer compression is used.

  – **FileName**. Specifies the name of the trace file to be written.
  – **OverWrite**. Identifies whether or not to overwrite the existing trace file with the same name specified by FileName. If disabled, each load will append the audit information to the same file separated by hashmarks, #####.
  – **AppendUniqueId**. Identifies whether or not to append a unique id to the end of the trace file name. As long as this is checked, a file with a unique name will be created and written to.

- **BufferedIoOptions**. Identifies a logical grouping of attributes used to define the behavior of the BufferedIO feature. (This is an advanced reading feature that reads data using a separate thread.)

Once you set the BufferedIOOptions parameter and run the process flow with the FileNetP8Loader, an internal check, the AdvancedIOThread, looks ahead for the datastreams and buffers and waits until it finds them. The log entry, "AdvancedIOThread waited four times for streams. AdvancedIOThread waited 0 times for buffers" appears when a preset time interval is reached and they have not arrived. As each datastream and buffer arrive, the AdvancedIOThread moves ahead looking for the next ones until all data is processed.

- **IsActive**. Designates whether the Buffered IO option is enabled. If this check box is selected, then the **Compression** parameter should be set to **NONE**.
- **BufferCount**. Specifies the number of buffers available for reading. The number must be at least one although much more is recommended. Different environments, network speed and document sizes being loaded all affect the optimal count. Benchmark tests are recommended.
- **BufferSize**. Specifies the size of the IO buffers. Any value over 0 is valid. A larger buffer size is recommended, although the sizes shouldn't be much over the average size of the documents being loaded. In other words, data in any given buffer will not be from more than one document, so a super large buffer will have much wasted space.

- **CommitIntervalOptions**. Identifies a logical grouping of attributes to specify the behavior of the batching feature.
  - **IsActive**. Specifies whether the batching feature is to be used.
  - **CommitInterval**. Specifies the maximum amount of documents that will be loaded before a commit is done.
  - **RollbackEnabledOnError**. Designates whether a rollback attempt occurs when an error is encountered during the load process. Both this and the IsActive parameter need to be enabled for a rollback attempt to take place.

    If set to **False**, a rollback attempt may still happen depending on properties set in the ReliableTransfer parameter.
- **Compression**. Contains settings related to the type of compression to use when loading. Note that compression cannot be performed when the BufferedIoOptions feature is active.
  - **Compression**. Designates the type of compression to use when loading. You can choose from:
    - **None**. No compression is performed.
    - **Zip**. Compresses individual files or groups of files with .zip extension.
    - **Gzip**. Compresses individual files with .gz extension.

    Note: If **BufferedIoOptions** > **IsActive** is enabled, then compression cannot be performed.
  - **Level**. Specifies the level of compression to use with the above setting.

    - **Default**. Default compression level.
    - **Best Speed**. Compression is optimized for speed.
    - **Best Compression**. Compression is optimized for size.
  - **CompressionProperty**. Indicates the property to use to indicate the compression type that was used when loading the document. If this is set, documents uploaded to FileNet will have this property which contains information about the type and level of compression used.
- **VersioningOptions**. See "FileNetP8Loader Versioning" on page 118.
- **AdvancedDataHandling**. Contains settings for dealing with data.
  - **UseEmptyValues**. Designates that any string properties defined in the index but have empty values are passed through to the FileNet instance as an empty string with the following default values:
    - string=no value
    - boolean=false
    - long=0
    - double=0.0
    - current date as the default for any date types

    If cleared, the FileNetP8Loader ignores any properties with no values and they will not be included in the update. In these cases, FileNet will use the defaults it has defined in the property template for that property within your FileNet instance.

    When enabled and used with versioning, the FileNetP8Loader defaults are used as the search values. When this parameter is disabled but versioning is enabled, the FileNetP8Loader attempts to retrieve defaults from the FileNet instance and if no defaults can be found, **IS NULL** is used to attempt to match records.
- **WaspLocation**. Indicates the location of the FileNet P8 WASP web services interface stores configuration file. This can generally be found in your FileNet P8 installation (for example, `C:\Program Files\FileNet\AE\CE_API\wsi`).
- **ReliableTransfer**. See "ReliableTransfer" on page 278.

5.2.5.1.2.6 FileNetP8Loader Versioning

When loading documents with the **FileNetP8Loader** into FileNet P8 you can choose to use document versioning. FileNetP8Loader versioning leverages the FileNet P8 versioning and a new document version is created when a document is checked in. Consideration should be taken before implementing this option as it increases the disk space required and will also impact the processing speed.

All the following parameters can be found under the **VersioningOptions** parameters for the FileNetP8Loader component.

Activating Versioning

Select the **IsActive** parameter to enable version control. When documents load, the search criteria will be checked for matching documents on the FileNet P8 server. Matching documents will be incremented (see the VersionType parameter below). Unmatched documents will be given a base version number (0.1 or 1.0). No documents are overwritten during this process.

If IsActive is cleared, versioning is not performed. Uploaded documents will simply load with their existing filenames. No existing documents are overwritten, even if there are duplicate filenames.

Note: Be aware that performance is greatly impacted when using the Versioning feature.

Version Incrementation Types

The **VersionType** parameter specifies the version incrementation type to use. You can choose from:

- **Major**. Increments the version number by a whole number (1.0, 2.0, 3.0, …).
- **Minor**. Increments the version number by a decimal number (1.1, 1.2, 1.3, 2.0, 2.1, …).
- **Limit**. Increases the version number by minor increments until the number of versions specified by the Limit parameter is reached, then the version increments by a major increment. For example, if the limit is set to 3, the increment pattern will be as follows: 0.1, 0.2, 0.3, 1.0, 1.1, 1.2, 1.3, 2.0... .

The **Limit** parameter indicates the number of minor revisions that must be present before a major version is checked in. To use this parameter, versioning must be enabled and the VersionType parameter set to Limit. (See the Limit option for the VersionType parameter for more information.)

Searching for Matching Documents

The following parameters can be used to search for documents within your database:

SearchProperties	Lists the specific search properties you want to use to determine a document match. The values listed must exactly match the values listed in the index file. If all of the listed properties are not a perfect match, then the document will be loaded as a new document. The Property names that are added should exist in all of the FileNet document classes being loaded to. If the property listed does not match an index property name then it will not be used. Furthermore, if no properties are added to the list, all the user defined properties located in the index file are used instead to query for matching documents.
	
	Enter the properties as they appear in the index file. For example:
	
	`GROUP_FIELD_NAME:AccountNumber`
	
	`GROUP_FIELD_VALUE:1111111111`
	
	Note: If a user has **RemoveSpaces** selected in the FileNetP8Loader configuration then spaces will be removed from all of the search properties.

IncludeFilename	Indicates whether to include the file name when making a document match. If selected, all properties, plus the file name must match to increment the version. If one property or the file name is not an exact match, the document will be loaded as a new document.
IncludeMimeType	Indicates whether to include the Mime type when making a document match. If selected, all properties, plus the Mime type must match to increment the version. If one property or the Mime type is not an exact match, the document will be loaded as a new document.

ContentOnly	Designates whether if only the content is updated in the creation of the new version, while the properties are left as-is when enabled. If cleared, the content and the properties are updated to create the new version of the document. This option will increase processing time. If a match is found, the existing document will be checked out and the new document will be loaded and checked in with an incremented version number.

Reverting to Previous Versions

Automatic rollback can be enabled to automatically be invoked on error by the following methods in the FileNetP8Loader properties:

- Under **CommitIntervalOptions**, select **RollbackEnabledOnError** and **IsActive**. This ensures a rollback will be done if there is an error while loading the batch.
- ReliableTransfer has a PerformCleanup option, which will invoke a rollback if Reliable Transfer does not enable the entire batch to run.

To use these rollback features with versioning, you need to add the property `XESVersionedBatchID` to your document class.

Note: These parameters must be enabled during component configuration, before the job runs. They cannot be invoked manually after an error has occurred. If a load fails and neither of these options were enabled, manually remove documents based on the batchID.

Troubleshooting

- In an error-free FileNet system, there should only be one copy of a document that matches the defined properties in the index file, whether the document is in a FileNet folder or stored as unfiled. This ensures the property search will only find one matching document during the upload and can perform the version incrementation. However, if a folder is deleted in the Workplace application of FileNet, the documents within the folder are not deleted and will remain saved as “unfiled” documents. If multiple batches have been uploaded into multiple folders and then some or all of these folders are deleted, there could be duplicate unfiled documents matching the index file search properties. If the FileNetP8Loader finds multiple documents, an exception will be thrown.
- If errors are encountered, use the reverting strategies mentioned above, which have been built into the FileNetP8Loader to remedy the situation. The best option is ReliableTransfer since this will only attempt a cleanup after all the retries are exhausted so performance should be better with this option. Failure

do use these cleanup features presents the risk of having documents being left in a checked out state.

5.2.5.1.2.7 FileNetP8Loader Supported Index Formats

The following section describes each supported index format.

Before the FileNetP8Loader uploads document objects, such as PDF, AFP, or Metacode batch files, metadata for these batch files must be defined in an index file. A mismatch between the properties of the index file and the document class will cause the FileNetP8Loader to fail when loading batch files.

Note: For more information, see "Indexing Parameters in the FileNetP8Loader Properties" on page 121 and "Preparing FileNet P8 for document loading" on page 108.

Indexing Parameters in the FileNetP8Loader Properties

The FileNetP8Loader component includes the following parameters to define an index file:

- IndexFile. The index file containing the paths to all the documents to load. This parameter also contains any document index data to include in the load.

  The FileNet P8 Loader component will accept two different types of index files:

  – OnDemand Index File Format

  – Proprietary Index File Format

  Note: Use the Index Writer component as part of an Output Transformation Project process to write an IBM Content Manager OnDemand or FileNet index file. You need to select the output file path you set in the Index Writer component as the file path in your FileNet P8 Loader component path. For more information on the Index Writer component, see *OpenText Embedded Output Transformation Engine - User Guide (VDTOTS-H-UTE)*.

  If you are using the IBM CMOD format, ensure that the IsIndexFormatIbmOnDemand parameter is set. For more information about IndexFile formatting, see the sample files in Index File Formats.

- IsIndexFormatIbmOnDemand. Indicates whether or not the index file is in IBM OnDemand format (compatibility mode).

Reserved Variables

Reserved variables are special attributes defined in an index and used by the FileNetP8Loader component. They have a unique name that starts with $. These attributes or properties are not part of the normal metadata and properties that are loaded with a particular document. They are used to allow flexibility in loading

documents. For example, $FILENAME would be used to reference the filename to be loaded into FileNet by the FileNetP8Loader.

Other reserved variables allow for dynamic changes, such as $DOCUMENTCLASS. The FileNetP8Loader will check to see if a document entry within the index is using the $DOCUMENTCLASS variable and will dynamically change the class for that document. If it does not find the $DOCUMENTCLASS variable for any particular document the DefaultDocumentClass specified in the FileNetP8Loader configuration file will be used.

The following are reserved variables used by FileNetP8Loader.

- `$FILENAME` – This variable is used to uniquely name the file as it is stored in the P8 archive. This should be uniquely set for each of the individual objects (logical documents).
- `$OBJECTSTORE` – This allows the ability to dynamically set the target object store in the FileNet P8 archive. If not specified for each object, the default object store defined in the FileNetP8Loader configuration will be used.
- `$MIMETYPE` – This allows the mime type to be dynamically set per object (logical document) if used. If not specified for each object, the default mime type set in the FileNetP8Loader configuration file will be used.
- `$FOLDERNAME` – This allows the target folder name to be dynamically set per object (logical document) if it is used.

  Note: It is recommended that you keep the **root folder** name **blank** or **/** if you plan to dynamically change ObjectStores because, in theory, you can use whatever RootFolder name they choose as long as that folder name exists with in all stores being used.

  If the name is not specified for each object, the default folder set in the FileNetP8Loader configuration file will be used.

  If the folder does not exist, it will be created.
- `$DOCUMENTCLASS` – This allows the target document class to be dynamically set per object (logical document) if used. If not specified for each object, the default document class set in the FileNetP8Loader configuration file will be used.

  The document class is the symbolic name that you define either in the index file or the FileNetP8Loader component.
- `$RESOURCEGROUPDOCUMENTCLASS` – This allows the target resource group document class to be dynamically set per object (logical document) if used. If not specified for each object, the default resource group document class set in the FileNetP8Loader configuration file will be used.

  The resource group document class is the symbolic name that you define either in the index file or the FileNetP8Loader component.
- `$RESOURCEGROUPKEYNAME` – This defines the resource document class property name that is set per object (logical document) if used. If not specified for each

object, the default resource group key name set in the FileNetP8Loader configuration file will be used.

- `$RESOURCEGROUPKEY` – This defines the resource document filename in FileNet that is set per object (logical document) if used. If not specified for each object, the default resource group key set in the FileNetP8Loader configuration file will be used.
- `$RESOURCEGROUPPATH` – This defines the resource file location locally (path and filename) that is set per object (logical document) if used. If not specified for each object, the default resource group path set in the FileNetP8Loader configuration file will be used.
- `$RESOURCEGROUPFOLDER` – This defines the uploaded resource file location (folder) on the FileNet server that is set per object (logical document) if used. If not specified for each object, the default resource group folder set in the FileNetP8Loader configuration file will be used.
- `$DOCTYPE` – This defines the document type property value (defined by `$DOCTYPEPROPERTY`). The property belongs to the resource associated document class set per object (logical document) if used. If not specified for each object, the document type set in the FileNetP8Loader configuration file will be used.
- `$DOCTYPEPROPERTY` – This defines the document type property name used by Output Transformation Server. The property belongs to the resource associated document class that is set per object (logical document) if used. If not specified for each object, the default document class set in the FileNetP8Loader configuration file will be used.
- `$BATCHIDPROPERTY` – This defines the BatchIdProperty name. The property is used by Output Transformation Server and belongs to the associated document class that is set per object (logical document) if used. If not specified for each object, the default document class set in the FileNetP8Loader configuration file will be used.

Sample Index file for using reserved variables

```
GROUP_FIELD_NAME:$OBJECTSTORE
 GROUP_FIELD_VALUE:ObectStore2
 GROUP_FIELD_NAME:$FOLDERNAME
 GROUP_FIELD_VALUE:XACard/P8Loader_Folder/test meta
 GROUP_FIELD_NAME:$MIMETYPE
 GROUP_FIELD_VALUE:application/meta
 GROUP_FIELD_NAME:$DOCUMENTCLASS
 GROUP_FIELD_VALUE:Test
 GROUP_FIELD_NAME:$RESOURCEGROUPDOCUMENTCLASS
 GROUP_FIELD_VALUE:Resource_P8Loader_XACard_DocClass
 GROUP_FIELD_NAME:$RESOURCEGROUPKEYNAME
 GROUP_FIELD_VALUE:Resource_P8Loader_XACard_DocProp
 GROUP_FIELD_NAME:$RESOURCEGROUPKEY
 GROUP_FIELD_VALUE:Resource_P8Loader_XACard_DocFilename
 GROUP_FIELD_NAME:$RESOURCEGROUPPATH
 GROUP_FIELD_VALUE:C:\XACard\P8LoaderDocClassChanges\DocPath.iores
 GROUP_FIELD_NAME:$RESOURCEGROUPFOLDER
 GROUP_FIELD_VALUE:XACard/P8Loader_XACardFolder
 GROUP_FIELD_NAME:$DOCTYPE
 GROUP_FIELD_VALUE:meta
 GROUP_FIELD_NAME:$DOCTYPEPROPERTY
```

```
 GROUP_FIELD_VALUE:XES_DOCTYPE
 GROUP_FIELD_NAME:$FILENAME
 GROUP_FIELD_VALUE:XAVIER ALISTAIR CAPLAN
 GROUP_FIELD_NAME:$BATCHIDPROPERTY
 GROUP_FIELD_VALUE:XA BATCH ID
 GROUP_FIELD_NAME:ACCOUNTNUMBER
 GROUP_FIELD_VALUE:XA123456
 GROUP_FIELD_NAME:CLAIMNUMBER
 GROUP_FIELD_VALUE:XAClaim123456
 GROUP_OFFSET:399822
 GROUP_LENGTH:200703
 GROUP_FILENAME:XA123.mta
```

OnDemand Index file format

Below are sample files, with descriptions, of the main indexing format to follow for the FileNetP8Loader component. Although any application can create the index file in this format, it follows the format of the IBM Content Manager OnDemand Index File format for those who may be familiar with it. It is the easiest format to use. One sample is for non-stacked files; the other is for stacked files.

Sample OnDemand Index File Format for Non-Stacked Files

This is the easiest method to use for loading non-stacked files (individual objects) from disk. This format is natively supported in Embedded Output Transformation Engine's **Indexer** and **Index Writer** components if using Output Transformation Engine for pre-document conversion from one print stream format to another.

```
COMMENT: OnDemand Generic Index FileFormat
 COMMENT: This File Has Been Generated By The Xenos Process
 COMMENT: Started Apr 6, 2009 2:11:19 PM
 COMMENT:
 CODEPAGE:500
 GROUP_FIELD_NAME:Customer_Name
 GROUP_FIELD_VALUE:April Black
 GROUP_FIELD_NAME:Policy_Number
 GROUP_FIELD_VALUE:01234-5512-DE
 GROUP_FIELD_NAME:Letter_Date
 GROUP_FIELD_VALUE:April 1, 2001
 GROUP_FIELD_NAME:Agent_Name
 GROUP_FIELD_VALUE:William Shakespeare
 GROUP_FIELD_NAME:$FILENAME
 GROUP_FIELD_VALUE:April Black-01234-5512-DE-pdf
 GROUP_OFFSET:0
 GROUP_LENGTH:0
 GROUP_FILENAME:April Black-01234-5512-DE.pdf
 GROUP_FIELD_NAME:Customer_Name
 GROUP_FIELD_VALUE:Chris Davis
 GROUP_FIELD_NAME:Policy_Number
 GROUP_FIELD_VALUE:34567-7734-GA
 GROUP_FIELD_NAME:Letter_Date
 GROUP_FIELD_VALUE:April 1, 2001
 GROUP_FIELD_NAME:Agent_Name
 GROUP_FIELD_VALUE:Mark Twain
 GROUP_FIELD_NAME:$FILENAME
 GROUP_FIELD_VALUE:Chris Davis-34567-7734-GA-pdf
 GROUP_OFFSET:0
 GROUP_LENGTH:0
 GROUP_FILENAME:Chris Davis-34567-7734-GA.pdf
 GROUP_FIELD_NAME:Customer_Name
 GROUP_FIELD_VALUE:Edward Ford
 GROUP_FIELD_NAME:Policy_Number
 GROUP_FIELD_VALUE:45678-9956-IN
```

```
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Edward Ford-45678-9956-IN-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Edward Ford-45678-9956-IN.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:George Harris
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:78901-2278-KS
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:George Harris-78901-2278-KS-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:George Harris-78901-2278-KS.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Ian Jackson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:67890-4489-MS
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Ian Jackson-67890-4489-MS-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Ian Jackson-67890-4489-MS.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Kevin Larson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:89012-0611-OK
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Kevin Larson-89012-0611-OK-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Kevin Larson-89012-0611-OK.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Michelle Nelson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:12345-8813-RI
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Michelle Nelson-12345-8813-RI-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Michelle Nelson-12345-8813-RI.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Oliver Perry
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:77777-1015-TX
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Oliver Perry-77777-1015-TX-pdf
```

```
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Oliver Perry-77777-1015-TX.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Quincy Robinson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:90123-3317-WY
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Quincy Robinson-90123-3317-WY-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Quincy Robinson-90123-3317-WY.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Sarah Thomas
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:23456-1219-AL
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Sarah Thomas-23456-1219-AL-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Sarah Thomas-23456-1219-AL.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Uma Vanderbilt
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:09876-6721-CT
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Uma Vanderbilt-09876-6721-CT-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Uma Vanderbilt-09876-6721-CT.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:William Xavier
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:42234-1023-UK
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Charles Dickens
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:William Xavier-42234-1023-UK-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:William Xavier-42234-1023-UK.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Ysadora Zbignew
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:76543-2324-DL
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Charles Dickens
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Ysadora Zbignew-76543-2324-DL-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:Ysadora Zbignew-76543-2324-DL.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:José Mañana
GROUP_FIELD_NAME:Policy_Number
```

```
GROUP_FIELD_VALUE:04551-8810-BR
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Charles Dickens
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:José Mañana-04551-8810-BR-pdf
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:José Mañana-04551-8810-BR.pdf
```

Description of the file is as follows:

- **Comment**. These are comment lines only and ignored by FileNetP8Loader.
- **Codepage**. A required entry of the index file, typically this is set to **500** and is only defined once directly under the `COMMENT` record(s).
- **Group_Field_Name**. This is one of the document class's index fields. The value is the **symbolic name** defined in the document class you are loading.
- **Group_Field_Value**. This is the index value to populate for the `GROUP_FIELD_NAME` defined directly above for the document being loaded.

  Note: Any number of `GROUP_FIELD_NAME` and `GROUP_FIELD_VALUE` can be defined for a given document but that property must be defined in P8.
- **Group_Offset**. For non-stacked file loads where individual objects will be loaded, this value is always **0**.
- **Group_Length**. For non-stacked file loads where individual objects will be loaded, this value is always **0**.

  Note: Special reserved variables (defined above starting with a "$") can be used as in this example where $FILENAME is used to uniquely name the document that is loaded into P8.
- **Group_Filename**. This is the name of the individual (logical file) on disk the FileNetP8Loader is loading. In the case of non-stacked files, this value will always be the physical file name on disk.

Sample OnDemand Index File Format for Stacked Files

This is the easiest method to use for loading stacked files (individual objects from a single physical file) from disk. This format is natively supported in the Indexer and Index Writer components if using Output Transformation Server for pre-document conversion from one printstream format to another.

```
COMMENT: OnDemand Generic Index FileFormat
 COMMENT: This File Has Been Generated By The Xenos Process
 COMMENT: Started Aug 6, 2009 7:08:24 AM
 COMMENT:
 CODEPAGE:500
 GROUP_FIELD_NAME:Customer_Name
 GROUP_FIELD_VALUE:April Black
 GROUP_FIELD_NAME:Policy_Number
 GROUP_FIELD_VALUE:01234-5512-DE
```

```
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:April Black-01234-5512-DE-pdf
GROUP_OFFSET:0
GROUP_LENGTH:10358
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name GROUP_FIELD_VALUE:Chris Davis
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:34567-7734-GA
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Chris Davis-34567-7734-GA-pdf
GROUP_OFFSET:10358
GROUP_LENGTH:13490
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Edward Ford
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:45678-9956-IN
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Edward Ford-45678-9956-IN-pdf
GROUP_OFFSET:23848
GROUP_LENGTH:10267
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:George Harris
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:78901-2278-KS
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:George Harris-78901-2278-KS-pdf
GROUP_OFFSET:34115
GROUP_LENGTH:10242
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Ian Jackson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:67890-4489-MS
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Ian Jackson-67890-4489-MS-pdf
GROUP_OFFSET:44357
GROUP_LENGTH:10224
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Kevin Larson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:89012-0611-OK
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Kevin Larson-89012-0611-OK-pdf
GROUP_OFFSET:54581
```

```
GROUP_LENGTH:10250
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Michelle Nelson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:12345-8813-RI
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Michelle Nelson-12345-8813-RI-pdf
GROUP_OFFSET:64831
GROUP_LENGTH:10512
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Oliver Perry
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:77777-1015-TX
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Oliver Perry-77777-1015-TX-pdf
GROUP_OFFSET:75343
GROUP_LENGTH:10263
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Quincy Robinson
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:90123-3317-WY
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Quincy Robinson-90123-3317-WY-pdf
GROUP_OFFSET:85606
GROUP_LENGTH:10403
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Sarah Thomas
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:23456-1219-AL
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Mark Twain
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Sarah Thomas-23456-1219-AL-pdf
GROUP_OFFSET:96009
GROUP_LENGTH:10209
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Uma Vanderbilt
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:09876-6721-CT
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:William Shakespeare
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Uma Vanderbilt-09876-6721-CT-pdf
GROUP_OFFSET:106218
GROUP_LENGTH:13906
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:William Xavier
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:42234-1023-UK
```

```
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Charles Dickens
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:William Xavier-42234-1023-UK-pdf
GROUP_OFFSET:120124
GROUP_LENGTH:10410
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:Ysadora Zbignew
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:76543-2324-DL
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Charles Dickens
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:Ysadora Zbignew-76543-2324-DL-pdf
GROUP_OFFSET:130534
GROUP_LENGTH:10377
GROUP_FILENAME:stackedPdf.pdf
GROUP_FIELD_NAME:Customer_Name
GROUP_FIELD_VALUE:José Mañana
GROUP_FIELD_NAME:Policy_Number
GROUP_FIELD_VALUE:04551-8810-BR
GROUP_FIELD_NAME:Letter_Date
GROUP_FIELD_VALUE:April 1, 2001
GROUP_FIELD_NAME:Agent_Name
GROUP_FIELD_VALUE:Charles Dickens
GROUP_FIELD_NAME:$FILENAME
GROUP_FIELD_VALUE:José Mañana-04551-8810-BR-pdf
GROUP_OFFSET:140911
GROUP_LENGTH:10333
GROUP_FILENAME:stackedPdf.pdf
```

The description of the file is as follows:

- **Comment**. These are comment lines only and ignored by FileNetP8Loader.
- **Codepage**. A required entry of the index file, typically this is set to **500** and is only defined once under the COMMENT record(s).
- **Group_Field_Name**. This is one of the document class's index fields. The value is the **symbolic name** defined in the document class you are loading.
- **Group_Field_Value**. This is the index value to populate for the `GROUP_FIELD_NAME` defined directly above for the document being loaded.

  Note: Any number of `GROUP_FIELD_NAME` and `GROUP_FIELD_VALUE` can be defined for a given document but that property must be defined in P8.

- **Group_Offset**. This is the starting byte offset of the stacked file for the logical document being loaded.
- **Group_Length**. This is the actual byte count of the document being loaded.

  Note: Special reserved variables (defined above starting with a "$") can be used as in this example where `$FILENAME` is used to uniquely name the document that is loaded into P8.

- **Group_Filename**. This is the name of the physical file on disk the FileNetP8Loader is loading the documents from. In the case of a stacked file, this

name will be the same throughout the index file and the GROUP_OFFSET and GROUP_LENGTH fields will be used to extract and load the individual logical documents.

Proprietary Index file format

Below are samples of the OpenText Proprietary Index File Format for use with the FileNetP8Loader component. This format can be created by any document and index producer or custom indexing application by customers or the OpenText Professional Services Group. One sample is for non-stacked files; the other is for stacked files.

Sample Proprietary Index File Format for Non-Stacked Files

```
 // Comment records begin with //.
 //
 ////////////////////////////////////////////////////////////////////////////////// /
/ One loadName is used for audit purposes. It can be blank. // -
loadName=Test ////////////////////////////////////////////////////////////////////////
////////// //
 // Values are for the ObjectStore, mimeType and FolderName (subdir from parm's
rootFolder).
 //
 // These values typically don't change. Blank values use the defaults in the P8 Loader
 // configuration file.
 //
 -folderName=
 -objectStore=
 -mimetype=
 ////////////////////////////////////////////////////////////////////////////////////
 //
 // One or more document sets to load.
 //
 //     - Documents should have a unique filename.
 //     - Any number of propertyName and propertyValue pairs can be given.
 //     - sourceFilename, sourceOffset and sourceLength describe where the actual
document
 //       bytes are stored in loading a "stacked" file.
 // The file is relative to this DocList file or can be absolute.
 // The -add command adds a new Document to the list.
 //
 -filename=TestClaim1
 -propertyName=PolicyNum
 -propertyValue=10315975308
 -propertyName=InsuredFirstName
 -propertyValue=ALEXANDER
 -propertyName=InsuredLastName
 -propertyValue=CHURCHILL
 -propertyName=ClaimNum
 -propertyValue=5-654123
 -propertyName=IncidentDate
 -propertyValue=05/04/01
 -sourceOffset=0
 -sourceLength=0
 -sourceFilename=Claims_doc0.pdf
 -add
 ////////////////////////////////////////////////////////////////////////////////////
 //
 // Second and subsequent -add commands use the same values unless reset.
 //
 //     - If the propertyName(s) are the same and in the same order, they do not have
 //       to be respecified.
 //     - If any propertyValues change, they must all be specified.
 //     - sourceOffset and SourceLength revert back to defaults (0 and length of the
```

```
//       file), so they should be respecified if used.
//
-filename=TestClaim2
-propertyValue=10285265410
-propertyValue=GABRIELA
-propertyValue=NERUDA
-propertyValue=4-8523369
-propertyValue=07/29/01
-sourceOffset=0
-sourceLength=0
-sourceFilename=Claims_doc1.pdf
```

Sample Proprietary Index File Format for Stacked Files

```
// Comment records begin with //.
//
////////////////////////////////////////////////////////////////////////////////////////////
///////////
// One loadName is used for audit purposes. It can be blank.
//
-loadName=Test
////////////////////////////////////////////////////////////////////////////////////////////
///////////
//
// Values are for the ObjectStore, mimeType and FolderName (subdir from parm's
rootFolder).
//
// These values typically don't change. Blank values use the defaults in the P8 Loader
// configuration file.
//
-folderName=
-objectStore=
-mimetype=
////////////////////////////////////////////////////////////////////////////////////////////
///////////
//
// One or more document sets to load.
//
//    - Documents should have a unique filename.
//    - Any number of propertyName and propertyValue pairs can be given.
//    - sourceFilename, sourceOffset and sourceLength describe where the
//      bytes actual document are stored in loading a "stacked" file.
// The file is relative to this DocList file or can be absolute.
// The -add command adds a new Document to the list.
//
-filename=TestClaim1
-propertyName=PolicyNum
-propertyValue=10315975308
-propertyName=InsuredFirstName
-propertyValue=ALEXANDER
-propertyName=InsuredLastName
-propertyValue=CHURCHILL
-propertyName=ClaimNum
-propertyValue=5-654123
-propertyName=IncidentDate
-propertyValue=05/04/01
-sourceOffset=0
-sourceLength=166031
-sourceFilename=Claims_doc0.pdf
-add
////////////////////////////////////////////////////////////////////////////////////////////
//
//
// Second and subsequent -add commands use the same values unless reset.
//
//     - If the propertyName(s) are the same and in the same order, they do not have to
//       be respecified.
//     - If any propertyValue(s) change, they must all be specified.
```

```
 //     - sourceOffset and SourceLength revert back to defaults (0 and length of the
file),
 //       so they should be respecified if used.
 //
 -filename=TestClaim2
 -propertyValue=10285265410
 -propertyValue=GABRIELA
 -propertyValue=NERUDA
 -propertyValue=4-8523369
 -propertyValue=07/29/01
 -sourceOffset=166031
 -sourceLength=139423
 -sourceFilename=Claims_doc0.pdf
 -add
```

5.2.5.1.3 FileNetP8Adapter

The **FileNetP8Adapter** component allows you to retrieve documents from IBM FileNet P8. The FileNetP8Adapter is the second-generation version (V2) of the FileNetRepositoryAdapter.

Once configured, the repository adapter component enables the retrieval and transformation of native print streams, as well as enabling PDF size reduction through the ability to transform documents to AFP file format for storage.

Before using this adapter, you must complete the preconfiguration procedure corresponding to your application server.

FileNetP8Adapter Properties

This component includes standard repository adapter parameters.

The following parameters are set for the FileNetP8Adapter component:

<*PostProcesses*>	Configuration for ViewAs settings.
<*AllPostProcesses*>	Contains configuration settings for all the logically named post processes defined in the repository instance.
<*LogicalName*>	Designates the name of the Output Transformation Server runnable to handle the conversion.
<*EsRunnable*>	Indicates the name of the process flow or runnable to execute that will handle the conversion.
<*InputXDocName*>	Specifies the name of the input XDoc to use for your runnable.
<*ResourceXDocName*>	Denotes the name of the resource XDoc that the runnable is expecting.
<*VersionedAfpResourceName*>	Indicates the name to use when adding retrieved versioned AFP resources as a job variable.
<*OutputXDocName*>	Specifies the name of the output XDoc to use for your runnable.
<*ReturnType*>	Designates the file type the resulting output should be.

<*JobVars*>	Defines additional job variables to be used in the post process.
<*Name*>	Indicates the name of the job variable.
<*Value*>	Indicates the value of the job variable.
<*ConversionsFrom*>	Contains settings related to file type conversions.
<*ExtFrom*>	Denotes the file type you wish to convert from.
<*DefaultTo*>	Identifies the default method used to view this type of file. A blank value means that no conversion is performed.
<*ConversionsTo*>	Contains settings related to all possible output formats.
<*ExtTo*>	Denotes the file type you wish to convert to.
<*OrderedList*>	Identifies the logically named post processes (typically with variables) in an ordered list, which are tested sequentially. If a match is found, the conversion is performed.
<*Authentication*>	Contains options relating to user authentication.
<*UseEsAuthentication*>	Designates whether or not the Output Transformation Server authentication engine should be used for initial authentication of users.
<*FileNetP8ConnectionFactory*>	Sets the connection parameters required to connect to the FileNet P8 instance.
<*WaspLocation*>	Indicates the location of the FileNet P8 4.x WASP configuration file. This can generally be found in your FileNet P8 installation folder (for example, `C:/Program Files/FileNet/AE/CE_API/wsi`). This value is not required for FileNet P8 5.2+.
<*LogicalName*>	Specifies a unique name for this adapter. Adapters are specified in the `auth.conf` file in the `/lib/` folder in your Output Transformation Server install directory.
<*UriConnectionString*>	Specifies the connection string to your FileNet server. If you are using FileNet 5.2 jars, use `FNCEWS40MTOM`. For example: `http://host:9080/wsi/FNCEWS40MTOM`. If you are connecting to FileNet 3.5.2, this value may be null.
<*ApiConfigFile*>	Specifies the configuration file pertaining to the connection to a FileNet 3.5.2 system. The file required can be found in your FileNet install directory under the file name `WcmApiConfig.properties`. If you are connecting to FileNet 4.0.2, this value may be null.
<*DomainName*>	Indicates the domain name of the current FileNet P8 4.x instance.
<*ObjectStore*>	Each domain can have multiple object stores. This entry indicates which object stores will be searched.
<*BundlePropertyName*>	Indicates the field name in the document's class where the bundle property ID is stored. The ID is used to reference resources the document needs for transformation.
<*DateFormat*>	Specifies the format to use when parsing dates. This creates the appropriate date format to use in FileNet queries.
<*Unfiled Search*>	Determines whether or not to search for unfiled documents.

<*VersionStatus*>	Determines the type of version status query when retrieving documents. The default value is **CURRENT** (the latest document checked in).
<*CustomResourceHandler*>	Provides configuration for how resources are cached and whether versioning of AFP resources occurs.
<*MemoryCacheSizeMeg*>	Indicates the maximum memory available, in megabytes, to cache resource groups in memory. A soft reference is used so that the memory is freed if the system requires more memory. A value of 0 means that no memory caching occurs. The default setting is 30.
<*DiskCacheSizeMeg*>	Indicates the maximum amount of disk space, in megabytes, used to cache resource groups to disk. Cached resources may remain available over a long period of time. A value of 0 means that no disk caching occurs. The default setting is 300.
<*DoAfpVersioning*>	Indicates whether or not AFP resource versioning occurs on disk. By default, this is enabled.
<*AfpResourcePath*>	Designates the base file path directory to hold versioned AFP resources. If this path is concurrently used by another repository, they will share the same AFP Resources Versioner and will not conflict with each other. The default directory is `./afpVersionedResource`.

5.2.5.2 Content Navigator plug-in

The IBM Content Navigator (ICN) plug-in available in Output Transformation Server helps facilitate connections between Output Transformation Server and an IBM Content Navigator server. Once the connection is made, you are able to select and submit documents from within ICN to Output Transformation Server for transformation or other processing.

5.2.5.2.1 Installing the IBM Content Navigator plug-in

In addition to a licensed copy of Output Transformation Server installed on your system, you must also have the ICN plug-in distributable ZIP file, which contains the necessary resources to set up the ICN plug-ins. The file is named `OTS-integration-icn-<version>.zip` and can be found in the `subproducts` folder of your installation package. (When facilitating connections to Output Transformation Server, your ICN software and server components must already be installed on your machine.) It is recommended you extract the contents of the ZIP file before starting the set up process as you will need access to these files at various points.

Note: To locate the requested document in a folder, the ICN plug-in queries for documents based on the ICN search results. Due to the resources needed, it is highly recommended you configure enough folder fields for documents to be uniquely searchable to minimize retrieval processing times and server load.

Setting up the Environment for the ICN plug-in

There are several steps involved in setting up Output Transformation Server with access to Content Manager OnDemand over ICN:

1. "Step 1: Configuring the WebSphere Liberty Shared Library" on page 136

2. "Step 2: Installing the OTS Integration JAR" on page 137
3. "Step 3: Installing and Mapping the OTS Viewer" on page 138
4. "Step 4: Setting Up the Document Context Menu" on page 138
5. "Step 5: Configuring the Desktop" on page 139
6. "Step 6: Configuring the ICN plug-in Components" on page 140

Note: Version 2.0.3 of the ICN Administration Console was used as the test environment for this. If you are using a different version, menu options and button names may be slightly different.

Step 1: Configuring the WebSphere Liberty Shared Library

On your WebSphere Liberty application server, you must define a shared library that can be referenced by the main IBM ICN web application.

To configure the WebSphere Liberty library:

1. From the ICN plug-in distributable file, open the `lib` folder and copy all the JAR files in the folder, with the exception of the `OTS-integration-icn.jar` file, to a suitable location on your WebSphere Liberty server.
2. Launch the **WebSphere Integrated Solutions Console**. (Check with your Administrator for the correct URL, but it will be similar to `https://localhost:9043/ibm/console/` on your network.)
3. Navigate to **Environment** > **Shared Libraries** and then click **Server Scope**.
4. Click **New** to establish a new library.
5. Set the following values for your library:
   - **Name.** Specifies the name for your shared library.
   - **Description.** Designates a description to help you identify the libraries for the ICN plug-ins.
   - **Classpath.** Contains a list of all absolute paths to all JAR files copied to the WebSphere Liberty server during step 1. You must define each JAR on a separate line. For example:

     ```
     C:\libraries\OTSPlugin\axiom-api-1.2.13.jar
     C:\libraries\OTSPlugin\axiom-dom-1.2.13
     ```
6. **Save** your changes.
7. Navigate to **Applications** > **Application Types** > **WebSphere Enterprise Applications**.
8. Select the ICN web application by clicking **Navigator**.

   The ICN web application properties screen appears.
9. Next, select **Shared library references**.

10. In the first row containing **Navigator** in the list, enable the corresponding check box.
11. In the same row, click the **Reference shared libraries** button.
12. On the following screen, add the newly created shared library to the **Selected** list. When you are finished, click **OK**.

    The **Shared library references** screen reappears and your shared library is added as a library reference.
13. Repeat steps 10-12 for all rows containing the navigator web application.
14. When you are finished, click **OK** and save your changes. You may need to restart the navigator web application for the changes to take effect.

> **Note:** Any class not found exceptions that occur during the use of the ICN plug-ins may be the result of this shared library set up not being completed correctly. It is recommended you ensure these settings are correct.

Step 2: Installing the OTS Integration JAR

The OTS integration JAR file contains some resources that must be uploaded to the ICN Administration Desktop before a connection can be made.

1. From the ICN plug-in distributable file, open the `lib` folder and locate the `OTS-integration-icn.jar` file.
2. Copy `OTS-integration-icn.jar` to a suitable location on your WebSphere Liberty server and make a note of the JAR file's absolute path.
3. Launch the **ICN Administration Desktop**, click **Plug-ins**, and then **New Plug-in**.
4. Select **JAR file path** and as the value, enter the absolute path to the `OTS-integration-icn.jar` file. When you are finished, click **Load**.

   Following a successful upload, some additional configuration options display at the bottom of the screen.
5. You must complete the following fields:

   - **Server URL**. Specifies the address and port for the server where Output Transformation Server is deployed.
   - **JobRunner Username**. Indicates the user name to use for the OTS JobRunner. By default, this is set to **JobRunner**.
   - **JobRunner Password**. Indicates the password to use for the OTS JobRunner. By default, this is set to **JobRunner**.
   - **WorkQueue Plugin name**. Denotes the name of the OTS ICN plug-in to show in the feature pane.
6. Click **Save and Close**.

Your new plug-in settings are saved.

Step 3: Installing and Mapping the OTS Viewer

The OTS Viewer is a custom viewer and is required to display your ICN documents. After installing the viewer on your server, you must set some configuration options to forward your OTS JobRunner results to the viewer.

Forwarding results to the viewer can be accomplished by creating a new viewer map if you are creating a new ICN Desktop or by modifying an existing viewer map of an existing desktop that incorporates the ICN plug-in. The instructions below guide you through the creation of a new viewer map, but if you are adapting an existing viewer map you can use the same viewer map settings.

1. From the ICN plug-in distributable file, open the `navigator` directory and locate the `OTSViewer.jsp` file.
2. Copy `OTSViewer.jsp` to the context root location on your web server where Content Navigator is deployed.
3. Launch the **ICN Administration Desktop** and click **Viewer Maps**.
4. Expand the available viewer mappings and right-click **Default viewer map** from the list. From the context menu that appears, click **Copy**.

   The **New Viewer Map** tab appears.
5. In the **Name** field, enter **OTSViewerMap**.
6. Next, click **New Mapping**.
7. On this screen, set the value in the **Repository Type** dropdown menu to **All Repositories**.
8. Set **Viewer** to **OTSViewer**.
9. Enable the **All Mime Types** check box.
10. Click **OK**.

    The **New Viewer Map** screen reappears.
11. Your newly created viewer definition, OTSViewer, appears at the bottom of the table. Select **OTSViewer** and move it to the top of the table by clicking **Move up**.

    Click **Save and Close** to save all your changes.

Step 4: Setting Up the Document Context Menu

Next, you must define a document context menu that contains the OTS plug-in action that adds items to the OTS document queue.

1. Launch the **ICN Administration Desktop** and select **Menus**.

2. Right-click **Default document context menu** and from the context menu that appears, select **Copy**.

3. In the **Name** field, type in a name to identify your new content menu configuration.

4. In the **Available Actions** section, locate an action named **Add to <WorkQueue>**, with the <WorkQueue> variable representing the name of the WorkQueue you defined during the OTS plug-in configuration. Select the action and add it to the **Selected Actions** list by clicking the right arrow. You must also organize the actions in the list as the order in which they appear in this list is reflected in the context menu.

   Note: If you see **Add to null** or an older name instead of the name of your existing WorkQueue, you should restart the navigator application for the changes to take effect.

5. Click **Save and Close** to save all your changes.

Step 5: Configuring the Desktop

Finally, you must create and configure a desktop in the ICN Administration Desktop that utilizes the ICN plug-in.

Note: Before you begin this process, ensure that you already have set up a repository that can be used to authenticate your new desktop.

1. Launch the **ICN Administration Desktop** and select **Desktops**.

2. On the Desktops screen, select **New Desktop**.

   The **New Desktop** tab showing the General subtab appears.

3. In the **Name** field, type in a name for your desktop. The **ID** field is automatically filled in as you enter content into the Name field.

4. In the **Description** field, enter a description that can be used to help identify the new desktop.

5. In the **Authentication** section, select the repository that is used to authenticate the desktop.

6. In the **Desktop Configuration** section, under **Viewer Map** you must select the viewer mapping you created while setting up the OTS Viewer.

7. Next, you must provide the OTS plug-ins with access to your repositories. Switch to the **Repositories** subtab and from the **Available Repositories** list on the left, select all the repositories you want the desktop to be able to perform searches on.

8. Subsequently, you must add the OTS Plug-in tab to the desktop. Change to the **Layout** tab and in the **Displayed Features** list, select the **WorkQueueFeature** check box.

9. In the same section, ensure that the **Search** and **Browse** features have a **Default Repository** defined for them.

10. Change to the **Menus** tab and under the **Content Context Menus** section, in the **Document context menu** field select the document context menu configuration you previously created.

11. Click **Save and Close** to save your changes.

Step 6: Configuring the ICN plug-in Components

Once the ICN plug-in is installed in Content Navigator, you are now ready to configure the ICN plug-in components in Output Transformation Server. There are two ICN plug-in components available; **IcnRequestHandler** and **IcnService**. The IcnRequestHandler component handles requests from the ICN plug-in, while the IcnService component processes requests from the ICN request handler. You must configure the components using their parameters, which you can retrieve descriptions for by clicking on a parameter.

5.2.5.3 OnDemand Connectors

The topics included in this section discuss how to configure and deploy OnDemand connectors.

5.2.5.3.1 ODAdapter

The**ODAdapter** component enables access to content stored in IBM Content Manager OnDemand (CMOD) to allow document and image transformation.

The OnDemand Web Enablement Kit (ODWEK) is an optional component of IBM's Content Manager OnDemand application. This feature enables you to access data stored on an OnDemand server through the use of a web browser instead of specialized software.

ODAdapter Properties

This component includes standard repository adapter parameters. For more information, see "Standard Repository Adapter Parameters" on page 89.

The following parameters are set for the ODAdapter component:

- **PostProcesses**. Configuration for ViewAs settings.
  - **AllPostProcesses**. Contains configuration settings for all the logically named post processes defined in the repository instance.
    - **LogicalName**. Designates the name of the Output Transformation Server runnable to handle the conversion.
      - **EsRunnable**. Indicates the name of the process flow or runnable to execute that will handle the conversion.

- **InputXDocName**. Specifies the name of the input XDoc to use for your runnable.
- **ResourceXDocName**. Denotes the name of the resource XDoc that the runnable is expecting.
- **VersionedAfpResourceName**. Indicates the name to use when adding retrieved versioned AFP resources as a job variable.
- **OutputXDocName**. Specifies the name of the output XDoc to use for your runnable.
- **ReturnType**. Designates the file type the resulting output should be.
- **JobVars**. Defines additional job variables to be used in the post process.
  - **Name**. Indicates the name of the job variable.
  - **Value**. Indicates the value of the job variable.

– **ConversionsFrom**. Contains settings related to file type conversions.

○ **ExtFrom**. Denotes the file type you wish to convert from.

○ **DefaultTo**. Identifies the default method used to view this type of file. A blank value means that no conversion is performed.

○ **ConversionsTo**. Contains settings related to all possible output formats.

- **ExtTo**. Denotes the file type you want to convert to.
- **OrderedList**. Identifies the logically named post processes (typically with variables) in an ordered list, which are tested sequentially. If a match is found, the conversion is performed.

- **Authentication**. Contains options relating to user authentication.
  - **UseEsAuthentication**. Designates whether or not the Output Transformation Server authentication engine should be used for initial authentication of users.
- **OdConnectionFactory**. Sets the connection parameters required to connect to your ID instance using ODWEK.
  - **ServerNameOrIp**. Identifies the OnDemand server or IP address to connect to.
  - **Port**. Specifies the port number used to connect to the OnDemand server. For example: 1445. If the value is set to -1, direct mode is enabled where TCP/IP is not used.
  - **OdwekTrace**. Sets the trace level for OnDemand debugging with ODWEK trace levels ranging from 0 (default value) for none, to a value of 4 for high. The trace log itself can be found in the `OdwekHome/trace` folder and if frequently used, should be occasionally cleared.

- **OdwekHome**. Indicates the path to the ODWEK JARS. Typically also contains the ars* and icu* DLL files. For example: `C:/ODWEK`.
- **ODServerConfig**. Contains the minimum properties required to establish a server connection.
  - **SpecifySystemParameters**. Indicates whether the OnDemand server configuration properties should be used. If false, the properties within the default `arswww.ini` file are used.
  - **AfpViewer**. Specifies the conversion method for viewing AFP data. You can select from **Ascii**, **Html**, **Native**, **Pdf**, **Plugin**, or **Xenos**.
  - **LineViewer**. Specifies the conversion method for viewing Line data. You can select from **Ascii**, **Applet**, or **Native**.
  - **MetaViewer**. Specifies the conversion method for viewing Meta data. You can select from either **Native** or **Xenos**.
  - **Timeout**. Designates the amount of time, in milliseconds, before a timeout request. The default value is 0 milliseconds, which also deactivates this attribute.
  - **HitListTimeoutResponse**. Specifies the behavior that is expected when a timeout on a request occurs. You can select from **ReturnPartialResultSet** (default) or **ThrowError**.
  - **MaxHits**. Designates the maximum number of hits to display. The default setting is 500.
  - **AppletDir**. Designates the directory where the applets reside.
  - **Language**. Indicates the language in which messages are output. The default is English (ENU). Consult your IBM Content Manager OnDemand User Guide for the latest complete list of supported languages and their language codes.
  - **TempDir**. Indicates the directory where temporary files are stored.
  - **TraceDir**. Indicates the directory where the trace file is written to.
  - **AdvancedPropertiesFilename**. Specifies the file path location to a properties file that contains any other configuration properties a user wants to set.

5.2.5.3.2 Configuring Output Transformation Server to Connect to an IBM Content Manager OnDemand Server

Connections from OpenText Output Transformation Server to an IBM Content Manager OnDemand Server can be set up either through direct modification of the OnDemand Web Enablement Kit's (ODWEK) `arswww.ini` file, or through the establishment of certain values for the OdConnectionFactory and ODServerConfig parameters within the ODAdapter component. Additionally, there are some preparations that must be performed prior to the actual server configuration to ensure that the connection can be made. The following steps summarize the process:

- "Step 1: Checking the ODWEK Client" on page 143
- "Step 2: Adding the ODWEK Client's Classpath" on page 143
- "Step 3: Copying the JAR File" on page 144
- "Step 3a: Configuring a shared library on WebSphere Liberty" on page 144
- "Step 4: Setting up the Server Properties" on page 145
- "Step 5: Testing your Connection (Optional)" on page 147

Step 1: Checking the ODWEK Client

Ensure that your OnDemand Web Enablement Kit client is properly installed and you have the most recent fixes and patches applied. The minimum supported version of the ODWEK is 10.1. If you require the client or information on the most recent patches, refer to your OnDemand installation materials or the vendor's website.

Also, note the ODWEK client's installation location as you will need to know this for subsequent steps.

Step 2: Adding the ODWEK Client's Classpath

You need to provide the ODWEK's classpath to the Windows system so that it knows where to locate the OnDemand's DLL and other configuration files. This is done by adding a path to the list of Windows environment path variables.

Note: The following instructions are for Windows XP, however, the procedure for other versions of the Windows operating system will be similar. Follow these directions as a general guideline.

To add the path to the environments list:

1. Open the Windows **Control Panel**, and double-click **System**.

   The **System Properties** screen appears.

2. Select the **Advanced** tab.

3. Click **Environment Variables**.

   The **Environment Variables** dialog appears.

4. In the System Variables list, scroll down until you find **Path**. Select the row to highlight it, and click **Edit**.

   The **Edit System Variable** dialog appears.

5. In the **Variable value** field, add the full file path location of your ODWEK installation. There are already several values already populated in the field so you must make certain that a semicolon separates the current variable values from your newly added value. Once complete, click **OK**.

   The **Environment Variables** dialog reappears.

6. Click **OK** until you have exited to the main **Control Panel** screen.

Step 3: Copying the JAR File

Some required ODWEK JAR files must be copied to the Output Transformation Server folder.

Navigate to the `<ODWEK_HOME>\api` folder and locate the `ODApi.jar` file. Copy this file into the `<OTS_HOME>\custom\jars` folder.

Users of OnDemand versions 10.1.0.5 or 10.5.0.0 must also copy the respective Gson JAR file to their `custom\jars` folder. The JAR files are included with Content Manager OnDemand installations and are located in the `jars` subfolder under the OnDemand installation location.

For users with OnDemand version 10.1.0.5 or later, Gson version 2.8.1 is used and the corresponding file name is `gson-2.8.1.jar`, which is stored in the `<ODWEK_HOME>\jars` folder.

For users with OnDemand version 10.5.0.0 or later, Gson version 2.8.6 is used and the corresponding file name is `gson-2.8.6.jar`, which is stored in the `<ODWEK_HOME>\jars` folder.

Step 3a: Configuring a shared library on WebSphere Liberty

If you are using WebSphere Liberty, you must complete some additional steps to create a shared library, which configures class loading for the Gson JAR file.

Users of other application servers can proceed to "Step 4: Setting up the Server Properties" on page 145.

To configure the shared library on WebSphere Liberty:

1. In WebSphere Integrated Solutions Console, you must create a shared library that references the folder where the Gson JAR file was copied. When creating the shared library on the **Shared Libraries** > **New** screen, use the following settings:

   - **Name**. Specifies a name used to identify the shared library instance.
   - **Classpath**. Designates the full path location to the `custom\jars` folder where the Gson JAR file is stored.

2. Next, you must assign the shared library to the OTSServer application. Navigate to the **Enterprise Applications** > **OTSServer** > **Shared library references** > **Shared Library Mappings** screen and move the shared library you created in the previous step from the **Available** list to the **Selected** list.

   When you are finished, save your settings.

Step 4: Setting up the Server Properties

The server connection properties can be configured through one of two methods:

- By setting some parameter values in the ODAdapter's OdConnectionFactory parameters, and optionally, several ODServerConfig parameters.
- By setting several parameter values in the ODAdapter's OdConnectionFactory parameters, and modifying the default `arswww.ini` file that is automatically created by the ODWEK client after installation.

Regardless of the configuration option selected, you must configure all parameters in the OdConnectionFactory section in order to establish a connection to the server. To set the OdConnectionFactory parameter values in the ODAdapter:

1. In Output Transformation Designer, either open an existing or create a new instance of the **ODAdapter**. Access the list of the component's parameters and locate the **OdConnectionFactory** section.

2. You must set the values for the following parameters according to the settings on the OnDemand server you are connecting to:

   - **ServerNameOrIp** identifies the OnDemand server or IP address to connect to.
   - **Port** specifies the port used to connect to the OnDemand server. Usually, this is set to 1445.
   - **OdwekTrace** sets the trace level for OnDemand debugging. The ODWEK trace levels range from **0** for none to **4** for high. The trace log itself can be found in the `OdwekHome/trace` folder and if frequently used, should be occasionally cleared.
   - **OdwekHome** indicates the full file path to the ODWEK JARS. Typically, this folder also contains the ars* and icu* DLL files.

3. **Save** your settings.

To complete the OnDemand server configuration, select whether you are configuring the server through the ODServerConfig parameters or by modifying the `arswww.ini` file, and proceed to the appropriate section below.

If you are configuring the OnDemand server through the ODAdapterODServerConfig parameters:

1. In Output Transformation Designer, either open an existing or create a new instance of the **ODAdapter**. Access the list of the component's parameters and locate the **ODServerConfig** parameters.
2. The **SpecifySystemParameters** indicates whether the component's OnDemand server configuration properties should be used. Set this parameter to **true** to indicate to the server that settings are to be retrieved from the ODWEK configuration properties, otherwise the settings in the default `arswww.ini` file are applied.
3. The other parameters under **ODServerConfig** illustrate the configuration properties that can be used to customize your server settings. You must set the values for these parameters with the appropriate settings for your OnDemand server. However, some parameters, such as the ones for the various viewers, can be left with their defaults if they are not applicable to your application.
4. **Save** your settings.

Proceed to "Step 5: Testing your Connection (Optional)" on page 147.

If you are configuring the OnDemand server through direct modification of the `arswww.ini` file:

1. Navigate to *<ODWEK_HOME>*`\JAVA\bin`, locate the `arswww.ini` file and open the file for editing.
2. You must change a few variables with the correct information about your OnDemand server. The sample code below shows a snippet of the `arswww.ini` file's settings:

```
[@SRV@_ark1]
HOST=ODServer1
PROTOCOL=0
PORT=1445
.
.
[configuration]
TemplateDir=/templates
ImageDir=/images
AppletDir=/applets
TempDir=C:/ODWEK_HOME/tmp
CacheDir=C:/ODWEK_HOME/tmp/cache
```

   Note: ark1 is the name of the OnDemand server in this example.

   Particular attention needs to be paid to the text in bold as these must be modified to suit your own system settings.
3. After your new values are entered, save the `arswww.ini` file.
4. In Output Transformation Designer, either open an existing or create a new instance of the **ODAdapter**. Access the list of the component's parameters and locate the **ODServerConfig** parameters.

5. The **SpecifySystemParameters** selection indicates whether the OnDemand server configuration properties should be used. Set this value to **false** so that the settings in the `arswww.ini` file are used, otherwise the settings from the ODAdapter's ODServerConfig parameter settings are applied.

6. **Save** your settings.

Proceed to "Step 5: Testing your Connection (Optional)" on page 147.

Step 5: Testing your Connection (Optional)

Before running any jobs using the ODAdapter, it is highly recommended to test your connection properties to ensure that it is set up properly.

Testing your connection is a simple process, which can be done with the **RepositoryAdapterSampleClientV2**. You can either compile the Java sample code that is included, or add the RepositoryAdapterSampleClientV2 component to a process flow or event. (If you opt for the latter method, ensure that the component maps to existing documents or the process will fail.) For more detailed information about using the RepositoryAdapterSampleClientV2 for testing or a general overview of the component, see *OpenText Output Transformation Server - Developer's Guide (VDTOTS-H-PGD)*.

5.2.5.3.3 Packaging and Deploying IBM Content Manager OnDemand Web Enablement Kit Generic Transform Interface

Content Manager OnDemand uses its Generic Transform Interface to administer data transforms from Output Transformation Server through the Content Manager OnDemand Web Enablement Kit (ODWEK) API set. After making a connection to the OnDemand server with the ODWEK API, you can also search and retrieve content. There are a number of steps that need to be completed in order to allow your generic client to perform transformations through the Generic Transform Interface. A sample client is also provided for users who do not have their own.

There are a few steps involved in setting up Output Transformation Server with access to the Content Manager OnDemand Generic Transform Interface:

1. "Step 1: Updating the ODWEK PATH Environment Variable" on page 148
2. "Step 2: Creating an ODWEK Generic Transform Interface Deployment Component" on page 149
3. "Step 3: Configuring the ODWEK Generic Transform Interface Deployment Component" on page 149
4. "Step 4: Testing and Deploying the ODWEK Generic Transform Interface Deployment Component" on page 152
5. "Step 5: Running the Generic Client Interface" on page 152

Prerequisites

Prior to starting these configuration steps, it is assumed that you already have met the following requirements:

- Installation of ODWEK
- Access to a Content Manager OnDemand (CMOD) server (for more information on Content Manager OnDemand, see *IBM Content Manager OnDemand Guide*)
- Installation of Output Transformation Server running either locally or remotely with a project configured to perform transform on your data from your CMOD server (for more information on installing Output Transformation Server, see OpenText Output Transformation Server Installation Guide, and for info on creating projects, see "Process Flows" on page 63.)

Furthermore, the ODWEK integration JAR file must be deployed in order to make the connection between Output Transformation Server and your Content Manager OnDemand server. The actual `Xenos-integration-odwek.jar` file is stored in the `<install_home>\dev-studio\lib` folder. A pre-unzipped copy of the ODWEK integration JAR is installed with the product in the `<install_home>\config\template\Xenos-integration-odwek` directory. The directory structure for the integration JAR file beneath the top level `Xenos-integration-odwek` folder is shown below:

Directory	Description
`\bin`	Contains the files required to run the sample.
`\output`	Location where output is written to by default.
`\sample`	Includes the full path and code files for the sample client.

Note: Subsequent references to the ODWEK integration folder in this section will be shown as `<ODWEKinteg_home>` so you must replace it with the file path location according to your own configuration.

Once these prerequisites are met, you can proceed to the first step.

Step 1: Updating the ODWEK PATH Environment Variable

First, you must manually update your PATH environment variable to point to the file path location where you installed your OnDemand software. Consult the *Content Manager OnDemand Web Enablement Kit Implementation Guide* for your particular version for instructions on how to perform the update for your current operating system.

Step 2: Creating an ODWEK Generic Transform Interface Deployment Component

The ODWEK Generic Transform Interface Deployment component can be initially configured by manually creating the component.

To create the ODWEK Generic Transform Interface Deployment component:

1. On the **File** toolbar, click New,  .
2. In the **Select a File Type** dialog, expand the **Tools** folder, then select **ODWEK Generic Transform Interface Deployment** and click **Next**.
3. In the **Name and Location of New ODWEK Generic Transform Interface Deployment** screen, complete the following fields:

   - **Name**. Specifies the name used to identify your new component. The name should be descriptive of its intended function.
   - **File System**. Indicates the local or network directory or archive file where your resources are stored.
   - **Directory**. Designates the specific folder in your file system to save your new component.

   When you are done, click **Finish**.

   The ODWEK Generic Transform Interface Deployment component is created in the directory you defined, and is opened in a new tab in the Development window where you can set the parameters according to your requirements.

Step 3: Configuring the ODWEK Generic Transform Interface Deployment Component

The ODWEK Generic Transform Interface Deployment component's properties are displayed on a new tab in the Development window. The fields being configured here replace the need to manually configure the Generic Transform XML (`XenosGenericTransform.xml`), Generic Transform properties (`GenericTransform.properties`), and Generic Client Interface ARG (`xenos-integration-odwek.argfile`) files, which are stored in your `<ODWEK_home>\bin` directory.

Note: If you prefer, you can still configure these files manually. Instead of completing the fields on this tab, open the `\subfolders` folder of your installation package and there is an `OTS-integration-odwek-<version>.zip` file. Within the ZIP file are copies of the Generic Transform XML, properties, and interface ARG files. Extract the files from the zip and then open them for manual editing.

When inputting the property names, ensure they also include the actual TransformName from the XML file as a prefix for all transforms. For example, you could enter `TestJob.processToRun=production_projects/afp2pdf/processflow/runjob.xProcessFlow` for your processToRun property. If no properties can be located in the file matching the processToRun property name, then any defined defaults without the processToRun name prefix are used.

The top half of the screen shows the connection details for your OnDemand server and Output Transformation Server deployment, while the bottom half shows settings for any transforms you want to run.

On the component's properties tab, the following fields are available:

In the OnDemand Server pane:

OnDemand Server

- **Host**. Identifies the server name or IP address for the OnDemand server.
- **Port**. Indicates the port number for the CMOD server to query.
- **User**. Specifies the user name used to connect to the CMOD server.
- **Password**. Specifies the password that corresponds with the user name.

Search Criteria

- **Folder**. Indicates the name of the CMOD folder to query. You can either enter the folder name manually or select a folder name from your OnDemand server in the dropdown menu.
- **Criteria Name**. Specifies the criteria field to use for the sample query.
- **Criteria Value**. Specifies the value to search for with your sample query.
- **Output File**. Denotes the file path location for where the Generic Client writes the output file to.

ODWEK Jar Location

- **ODWEK Jar**. Indicates the location of your ODWEK API JAR file. By default, `ODApi.jar` is the file name. You must enter a valid JAR location in order to connect and send queries to your OnDemand server.

In the Output Transformation Server pane:

Output Transformation Server

- **Server URL**. Identifies the location of your Output Transformation Server connections. You can either enter the URL manually or click the **ellipsis** button,  , to browse a list of established connections to choose from. You must also indicate whether you want to use **Username/Password** or **Token** type authentication. Only the configuration fields for your selected authentication type are enabled.

- **User**. Specifies the user name to use for authentication.
- **Password**. Specifies the password that corresponds to the user name for authentication.
- **Token**. Indicates the token to use for authentication purposes. You must also set the token as the value for the AuthSchemeConfig parameter within the system configuration for the Output Transformation Server instance being accessed.

XDoc Mappings

- **Input Key**. Denotes the input mapping key. The default for this property as set in the Generic Transform properties file is input.
- **Result Key**. Denotes the output mapping key. The default for this property as set in the Generic Transform properties file is output.
- **Resource Key**. Denotes the resource mapping key. The default for this property as set in the Generic Transform properties file is result.

In the Process to Run pane:

- **Process to Run**. Displays a list of runnable components or process flows to run to perform the transform. You can add items to the list by clicking the **Add** button,  , or by dragging and dropping a runnable component or process flow file. Also, selecting an entry from this list populates the other fields in this pane with information, if available, about the component or process flow.
- **Override common configuration**. Denotes whether to override the base server and XDoc mapping properties. When enabled, you must also provide the XDoc mapping keys you want to use in the fields below.

Transform Properties

- **Transform Name**. Indicates the name of the transform. This name is used as the viewer argument and is passed to the ODWEK Retrieve APIs.
- **Description**. Denotes a description of your transform.
- **Mime Type**. Specifies the MIME type of the data as it is returned from the transform. Using the dropdown menu, you can select from **application/pdf**, **application/xml**, **text/plain**, or **text/html**.
- **Extension**. Specifies the extension of the data as it is returned from the transform.

Application Properties

- **Application Group**. Displays a list of available ODWEK application groups, which contain the index information necessary to load, search for, or retrieve reports.
- **Application Name**. Displays a list of available ODWEK application names, which contain metadata about your reports or documents.

XDoc Mappings

- **Input Key**. Denotes the input mapping key. The default for this property as set in the Generic Transform properties file is **input**.
- **Result Key**. Denotes the output mapping key. The default for this property as set in the Generic Transform properties file is **output**.
- **Resource Key**. Denotes the resource mapping key. The default for this property as set in the Generic Transform properties file is **result**.

Step 4: Testing and Deploying the ODWEK Generic Transform Interface Deployment Component

Once you have configured the component, you are ready to test and deploy the sample client. While testing is a completely optional procedure, it is highly recommended.

Clicking the **Test** button allows you to analyze whether a connection to the CMOD server can be made using your current settings. The **Choose Test Output Folder** dialog appears where you must select a folder to store the output files. If a connection cannot successfully be made, you must revisit your configuration settings to ensure they are correct for your server.

When you are ready, click **Deploy** to generates a ZIP file containing the necessary files to allow transactions through the sample client interface. You are prompted to choose a location to save the file.

Step 5: Running the Generic Client Interface

After completing the previous configuration steps and deploying the ZIP file, the Generic Client Interface is ready to bridge communications between Output Transformation Server and your CMOD server.

Note: If you are running your own generic client interface then performing the step below is not necessary and you can start your generic client, but you can carry them out if you wish to test your connection to ensure that the web service you are using is functional.

Navigate to the `<ODWEKinteg_home>\bin` folder and Windows users should run the `GenericClient.bat` file, while Linux users should run the `GenericClient.sh` file.

5.2.5.3.4 Configuring SSL connections with ODAdapter

To configure the ODAdapter component with SSL:

1. You must create a properties file (for example, `ssl.properties`) with the specified key value pairs using your preferred text editor. The properties file must contain the following:

   ```
   UseSSL=TRUE
   SSLKeyRing=<KDB_File_path>
   SSLKeyStash=<STH_File_path>
   ```

   where

   <KDB_File_path> is the file path to the keyring file.

   <STH_File_path> is the file path to the stash file.

2. In the `.xOnDemandRepoAdapterV2` file for your ODAdapter instance, locate the **OdConnectionFactory > ODServerConfig > AdvancedPropertiesFilename** parameter and for the value, enter the file path location to the properties file you created in the previous step.

3. Go to **OdConnectionFactory > port** and update the value with the SSL port number for your OnDemand server.

4. Save your ODAdapter changes.

5. Restart your Output Transformation Server instance.

5.2.5.4 IBM WebSphere MQ Series Connectors

IBM WebSphere MQ Series (simply known as WebSphere MQ) is a messaging queue, which runs within IBM WebSphere application server Liberty. The application can communicate with WebSphere MQ with the MQEvent and MQProcess components. These can either be run from Output Transformation Server **standalone** or from an Output Transformation Server instance deployed within an **application server** to implement an IBM WebSphere MQ Series component.

5.2.5.4.1 Communicating with IBM WebSphere MQ Series

Users that need to connect directly to a WebSphere MQ queue from Output Transformation Server can do so either from the **standalone** version or from Output Transformation Server running with an **application server** to implement a WebSphere MQ component.

This can be done either:

- **Locally.** The **MQProcess** and/or **MQEvent** components run on the same machine as the WebSphere MQ Server.
- **Remotely.** The MQProcess and/or MQEvent components run on a different machine from the WebSphere MQ Server.

Related Links

- "IBM WebSphere MQProcess" on page 228
- "IBM WebSphere MQEvent" on page 180

5.2.5.4.1.1 Setting up your Output Transformation Server Environment

To communicate with IBM WebSphere MQ Series:

1. The WebSphere MQ Server must be up and running, as well as MQManager.
2. Queues or Topics to deliver messages need to be created on WebSphere Liberty in advance.
3. Copy the following MQ libraries (JAR files located in `%WebSphereMqInstallDir%\Java\lib`) to your *<install_home>*`/lib/ext` directory:

   ```
   com.ibm.mq.jar
   com.ibm.mqjms.jar
   connector.jar
   dhbcore.jar
   fscontext.jar
   jms.jar
   providerutil.jar
   ```

   Once you add these files, you can create, modify, and execute the MQProcess and or MQEvent like all other components in Output Transformation Server by using wizards for setup.

   Note: If Output Transformation Designer is running when you copy in the JAR files, you will need to restart it.

4. If running from Output Transformation Designer, start Output Transformation Designer and create your components as you would anything else. If you or someone else has already created the components, simply mount the file system they already reside in.
5. If running from an application server, deploy the WebSphere MQ components to the application server running Output Transformation Server. For more information on deployment or packaging an `.ear` file, see "Deploying the Packaged Application to an Application Server" on page 303.
6. Execute the MQProcess or MQEvent.

   MQProcess sends data into the WebSphere MQ Server queue.

   MQEvent will watch the input queue and pass data to a process flow like other Output Transformation Server events.

5.2.6 Third Party Connectors - LRS

To assist in connecting with an LRS server, the VPSXPrintProcess process is available.

5.2.6.1 VPSXPrintProcess

The **VPSXPrintProcess** component initiates the VPSX Print Request process when it receives the print file and sends it to the VPSX print server for printing. Printing through the print server allows tracing of print jobs and also more security. The print server logs information such as owner, job status, size of file and a time stamp, among other data, prior to printing the document.

A third party Java Archive (`lrsipp 1.0.jar`) library is required in the Output Transformation Server classpath at runtime before the VPSXPrintProcess component can function. Without the library file, the component will not run. For more information on the Output Transformation Server classpath, see *OpenText Output Transformation Server - Developer's Guide (VDTOTS-H-PGD)*.

5.2.6.1.1 VPSXPrintProcess Properties

The following parameters are available for VPSXPrintProcess:

UsePrintFile	Defines whether the print file from the file system or from the input map is used. If true, the print file from the file system is used. If false, the print data from the input map is used.
VPSXPrintDest	Specifies the print destination of the job. This can be set statically or dynamically through job variables.
PrintFileName	Indicates the name of the print file, which can be set statically or dynamically through job variables. If specified statically, it is the absolute path of the print file. If specified dynamically, it is the value of the job variable.
PrintCopies	Indicates the number of copies for the print job.
VPSXServerName	Specifies the name of the VPSX server to connect to.
VPSXPort	Indicates the VPSX server port to use.
VPSXUserId	Specifies the user ID to connect with.

5.2.7 Third Party Connectors – Microsoft

The topics in this section describe the Microsoft Message Queue (MSMQ).

5.2.7.1 Overview of MSMQ Connectors

MSMQ is a Microsoft message queue utility.

The **MSMQEvent** watches for a file in the Microsoft message queue and then starts the related process flow.

The **MSMQProcess** uploads files to the MSMQ.

Related Links

- "MSMQEvent" on page 182
- "MSMQProcess" on page 230

5.2.7.1.1 About MSMQ

Current Microsoft operating systems include MSMQ, but it is not installed by default. Microsoft Windows Help provides full instructions to install and configure this component, based on your operating system and your network setup. To access these help documents:

1. Open **Windows Explorer**.
2. From the toolbar, choose **Help** > **Help and Support Center**.
3. Enter **MSMQ** in the Search field.
4. The document **Using Message Queuing** explains the MSMQ installation and configuration.

Alternatively, you can browse the Windows Help folder contents in the `C:\Windows\Help` folder.

Once MSMQ has been installed, you can view the message queue at **Control Panel** > **Administrative Tools** > **Computer Management** > **Services and Applications** (expand in the left pane) > **Message Queue** (expand in the left pane).

MSMQ supports Public and Private Queues. Each type can contain multiple queues. To create a new queue, right-click **Public Queues** or **Private Queues** and choose **New**.

Each message has a message name label, a body as message content and a correlation id.

5.2.7.1.2 Using MSMQ with Output Transformation Server

Output Transformation Server has two components that can interface with MSMQ.

- The **MSMQProcess** uploads files to the MSMQ.
- The **MSMQEvent** watches the MSMQ for files to be processed.

Before you can begin using the MSMQ components, take note of the following and follow the associated procedures if they relate to your configuration options:

- Ensure that the OpenText supplied `MsmqJava.jar` file is in the `<install_home>\lib\common` folder. If it is not, you may be running an older version of Output Transformation Server, or it has been removed. Contact OpenText Customer Support or your OpenText representative.
- If you are going to connect to remote MSMQ or deploy your project on IBM WebSphere Liberty you need to grant permission to the computer to access the Queue. This must be done even if WebSphere Liberty is installed on the same computer and you set MqHost to localhost.

1. From the **Control Panel**, choose **Administrative Tools** > **Computer Management** > **Services and Applications**.
2. Expand **Message Queueing** in the left pane and located the message queue.
3. Right-click on your message queue and choose **Properties**.
4. On the **Security** tab, click **Add** at **Group or user names**.
5. On the **Select Users**, **Computers**, or **Groups** form, click **Object Types**.
6. On the **Object Types** form, ensure that **Computers** is selected and click **OK**.
7. On the **Select Users**, **Computers**, or **Groups** form, choose the computer location and enter the computer name of the computer where the dll file resides (where Output Transformation Server is installed) and click **OK**.
8. Grant the applicable permissions based on your requirements.
9. Click **OK** and exit the **Control Panel**.

If you are using the Windows 64-bit platform you need to do the following:

1. Navigate to `<install_home>\lib\dll`.
2. Rename MsmqJava.dll to MsmqJava_32bit.dll.
3. Rename Msmq_64bit.dll to MsmqJava.dll.

Note: You may want to move the MsmqJava.dll file from its location in the file system `<install_home>\lib\dll` to the WINDOWS/System32 file system, or to the Java library (`C:\Java\lib`) so it is with the other system or Java dll files. If you create a new folder for this file, do not name the folder MsmqJava. Note the location of MsmqJava.dll.

You are now ready to configure an MSMQEvent or an MSMQProcess.

Related Links

- "MSMQEvent" on page 182
- "MSMQProcess" on page 230

5.3 Events

Events are service components that wait for a particular user-specified incident to happen on the system. They can observe a watched directory or FTP site for files to be written, listen for incoming email or JMS messages, or await data to be delivered through a network socket.

Events can be configured to be fully automated, as well. An event can be set to kickoff in conjunction with the initial startup of the engine by adding it to the **ServiceAutoStart** parameter in the system configuration. Also, **SystemSchedule** allows you to program jobs and services to automatically run when the engine starts.

Since an event alone cannot develop the data it receives, once an event picks up the data type it was waiting for, it triggers a process flow. Different events and processes can be combined together to produce your desired result. For example, an XML file uploaded to an FTP server can be manipulated in a number of ways to suit your requirements through a series of events and process flows. An FTP Event can be configured to run a process flow when an XML file is placed on an external FTP server. The XML data is read from the FTP server by the event, which automatically starts the process flow. The first process in the flow converts the XML into both PDF and CSV formats. The next process sends the PDF over a socket using the SocketProcess, while the final process in the flow writes the CSV to disk using the FileProcess. For more in-depth information on process flows, their configuration and usage, see "Process Flows" on page 63.

Some events and event processes can handle any size of data, whereas others are limited because the data must reside fully in memory. Also, some events require a response to be sent back as an acknowledgement. These details are discussed in subsequent sections.

Several samples of events are provided in the directory:

```
install_home\initialFiles\common\_sample\OutputTransformation\event
```

5.3.1 Event Tab

When an Event is initially opened in the Development window, you see a screen similar to the Process Flow tab in Development Window figure. The initial Event contains the **Event** icon by itself. A process flow can be dragged from either the File System or the Palette and dropped onto the editor.

- On the top is the **Event**. This is where you configure the type of event that will trigger the process flow. Double-click the icon in the box to display the Event properties.
- On the bottom is the **Process Flow**. This is where you configure the series of processes that will be triggered by the event. Double-clicking the gears icon in the Process Flow box opens a new tabbed pane displaying the process flow chart for this process flow. You can also open a process flow by selecting it in the File Systems tab.

A red X will display until the process flow is configured, at which time a blue arrow will display.

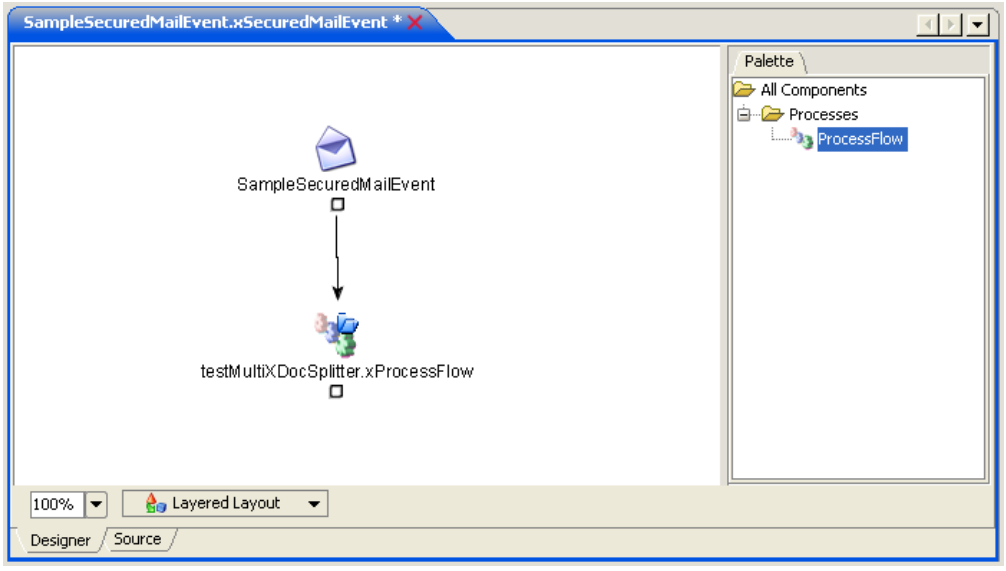


Figure 5-3: Event tab in Development window with a configured event

Related Links

- Figure 4-1
- "Configuring an Event" on page 161

5.3.1.1 Event Designer Tab

The **Designer** tab, located at the bottom left-hand side of the **Event** tab, displays the graphical representation of the Event. It is also where you build the Event by adding process flows.

From this tab you can configure the Event properties and the process flow properties by right-clicking the icons and selecting **Properties**.

5.3.1.2 Event Source Tab

As the process is built up, the XML source code for the process flow is created behind the scenes. To see the XML source code, switch to the **Source** tab, located at the bottom left side of the Event tab, and you are presented with an XML representation of the process flow as defined in the Process Flow tab.

5.3.1.3 Event Palette

On the left-hand side of the **Event** tab is the **Event Palette**. This palette is similar to the **New Component** dialog window, but it only shows process flow templates.

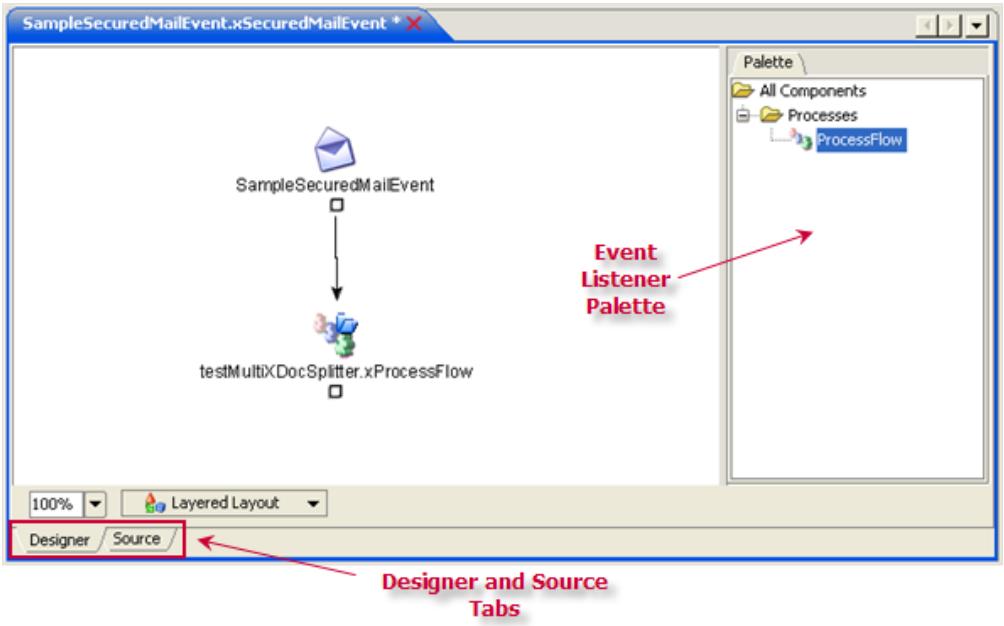


Figure 5-4: Designer and Source tabs, and Palette in the Event tab

5.3.2 Creating an Event

To create an event:

1. Right-click on the file system, and select **New**, then **Component**.

   The **New Component Wizard** opens with the **Select a File Type** dialog.

2. Expand the tree under **Events**.

3. Select an event type, and click **Next**.

   The **Event Wizard** or **Express Setup** screen is displayed.

4. You can either:

   - Select **Use Wizard**, which will launch a wizard that will guide you through the steps for configuring the event.
   - Select **Express Setup** to create the event with a name and location that you specify. After it is created, you can configure the event at a later time. For more information, see "Configuring an Event" on page 161.

5.3.3 Configuring an Event

To configure an event:

1. Drag a previously defined event from the **File Systems tab** and drop it on the Development window.

2. Right-click the event on the **Development window**.

   A context menu appears.

3. Select **Properties** from the context menu. The **Properties window** appears with the available parameters that can be set for the type of event you have chosen.

4. Set the following parameters (additional event-specific parameters may also be set):

SleepTime	This is set by using the <*MinimumSleepTime*> and <*StandardSleepTime*> parameters. Sleep time represents the polling frequency, or how frequently to check whether a new file has been added to a directory in the list of watched directories.
ProcessToRun	This is the process flow to run when the event finds the file it is listening for. This is set when you drag a process flow onto the **Event** editor.

5.3.4 Running an Event

To run an Event:

1. In the **File Systems** tab, locate the event that you want to run.
2. Right-click on the nt and select **Execute**, or highlight the event and click on the **Execute** icon, , in the **Build** toolbar.

   The **Execution Options** dialog is displayed.

   The event is automatically saved and **Execution Options** dialog opens.

   Note: You can use the **Undo** function to revert changes to the saved event.

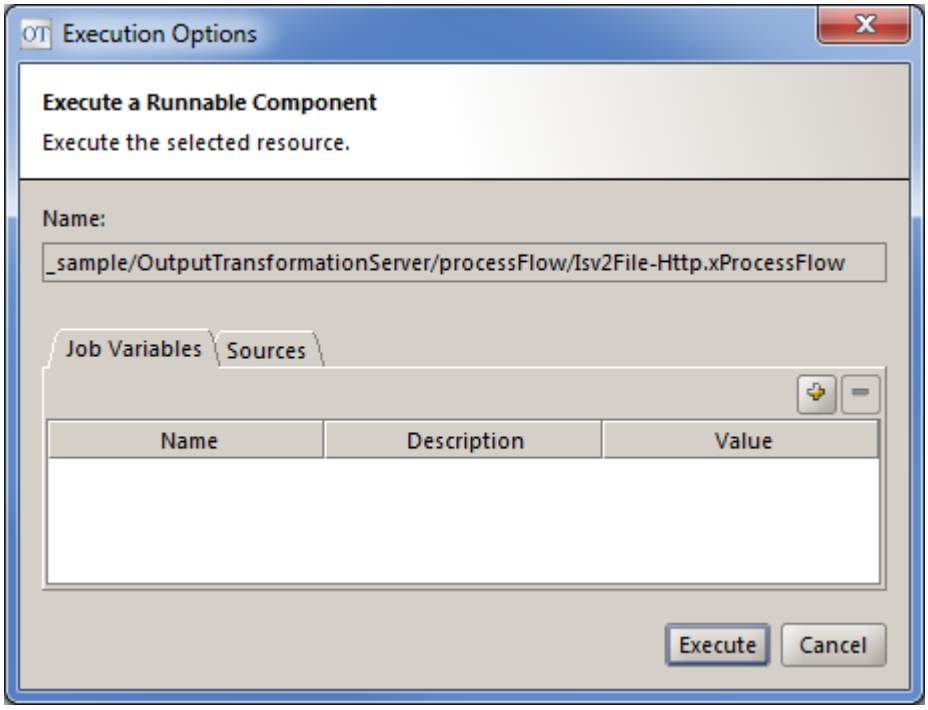


Figure 5-5: Execution Options dialog

3. The **Execution Options** dialog allows you to define the job variables required by the event. The following information is displayed:

   - The **Name** field indicates the path and name of the event to be executed.
   - The **Job Variables** tab is automatically populated with any job variables entered in the **ServiceAutoStart** parameter of the system configuration file. You can supply additional job variables or remove existing ones by clicking on the **Add**, , or **Remove**, , buttons. For more information on the **ServiceAutoStart** parameter, see "ServiceAutoStart Parameter" on page 21.

Define your job variables by entering a name, description, and value for each variable in the respective columns.

4. Click **Execute**.

   The event is executed and the **Output** tab appears in the **Events** window.

5.3.5 Stopping an Event

Stopping an Event will prevent your development machine from slowing down to scan for input files when you know there are no files to be found.

1. Select the running Event in the **File Systems** tab.

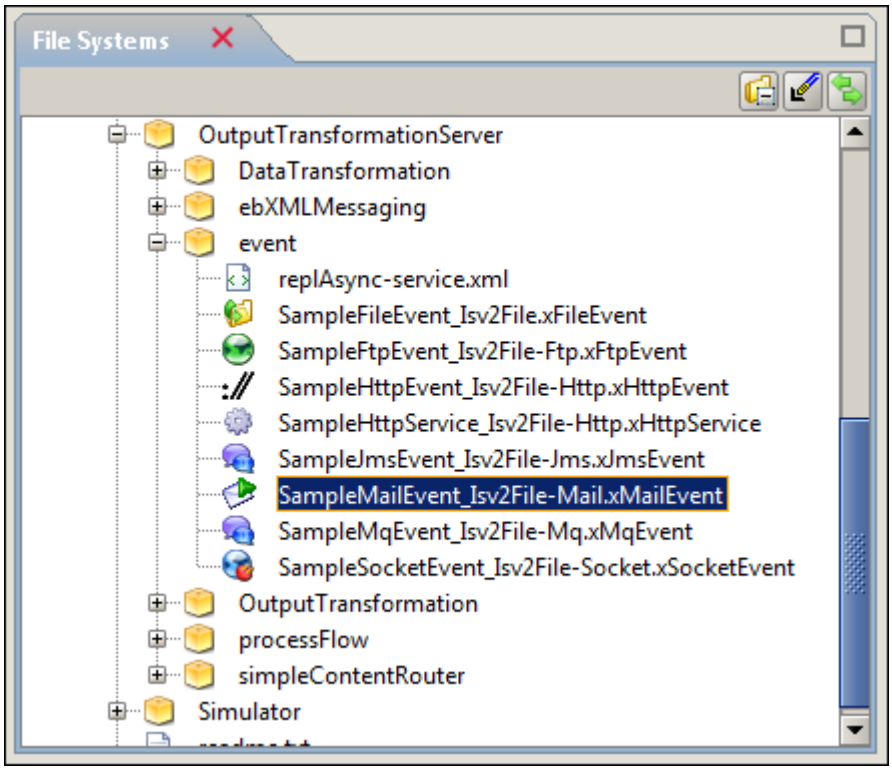


Figure 5-6: File Systems window with running event selected

2. Right-click the event and from the context menu that appears, select **Stop**. Alternatively, you can also click the **Stop** icon in the Events tab.

5.3.6 Running a Sample Event

To run a sample Event:

1. Select the Event in the **File Systems** tab.
2. In the **File Systems** tab, expand the subdirectories under *<install_home>* `\initialfiles\common` to show the files under `\_sample\ OutputTransformation\event`. Double-click **SampleFileEvent_Isv2File.xFileEvent** to open it in the Development window. You will see that the selected event is shown with a red circle (stop light). This indicates that this event is open in a Development window but it is not running; it is not currently looking for input files.
3. Click the **Execute** icon on the toolbar or right-click the event and select **Execute** from the menu. The event starts running, scanning the designated directory.

   When the event is active, the red circle changes to a green arrow, and a matching pane will open in the **Events Window**.
4. Copy a file to a watched directory. To see it in action:
   a. Minimize Output Transformation Designer.
   b. Open Windows Explorer.
   c. Browse to the following directory:

      `<install_home>\initialFiles\common\_sample\DataTransformation\csv`
   d. Copy `Input.csv` to the following directory:

      `<install_home>\sampleApplication\scanTest\`

      You will see the extension change when traffic detects and picks up the file.
5. Check the output in `\sampleApplication\testResult`, then maximize Output Transformation Designer to see the event log.

5.3.7 Event Types

The topics in this section discuss the types of events and events in general:

5.3.7.1 Standard Event Parameters

All events include the following standard parameters to define basic setup and function.

- **ProcessToRun**. Specifies the path and filename of the process flow to be run when the event is triggered. (Required, and may be added automatically by connecting the event to a process flow in Output Transformation Designer.)
- **InputMapName**. Allows for override of input data name (optional/rarely used).
- **InvocationDescription**. Specifies the description to appear in the JobStats indicating how the process flow was started; defaults to the name of the event configuration file.

- **Comment**. Specifies a comment to appear in the JobStats when the process flow is submitted. (Optional)
- **MaxSubmittedJobs**. Specifies the maximum number of inputs that can be processed at one time. If another is found, it must wait for one to finish before being processed.
- **JobVar**. These job variables will be initialized with the values before the process flow runs.
- **RestartOnError**. Determines restart behavior in the event of an abnormal service shutdown.
  - **Retries**. Indicates the number of attempts to restart the event before permanent shutdown occurs. The default value is 0 (no retries).
  - **TimeBetween**. Indicates how much time (in seconds) to wait before an attempt is made to restart. The default value is 60 seconds.
- **MinimumSleepTime**. Specifies the minimum amount of time to wait between scans for new input (from the end of one scan to beginning of the next), in milliseconds (defaults to 0).
- **StandardSleepTime**. Sets the standard amount of time from the beginning of one scan for new input to the beginning of the next scan, in milliseconds. (Required)
- **EventShutdownTimeout**. Specifies how much time (in seconds) to wait before forcing a shutdown of this event when a shutdown is requested. The default value of -1 indicates that event shutdown will be completed gracefully after all jobs started by the event were completed.

5.3.7.2 Clustering Parameters

All events include a section dedicated to clustering setup. The parameters are identical in all events and are as follows:

Singleton	Specifies whether or not to run the event as a singleton in the cluster. This can be useful in preventing duplicate processing when there are multiple versions of the same event running simultaneously.
LoadBalancing	Specifies whether or not to support cluster load balancing for the event. If enabled, messages will be distributed across the nodes within the cluster. If disabled, all messages will be processed on the master node.
JobFailOver	Specifies whether or not to support failover for jobs that are running when an event fails. If this option is selected, then if the server fails during message processing, received messages will not be deleted, and will be picked up again for reprocessing once the server is restarted.

5.3.7.3 EbXmlErrorMessageHandler

The **EbXmlErrorMessageHandler** component catches invalid ebXML messages received on the messaging server and initializes a predefined process flow to deal with the invalid messages.

For more information, see *OpenText Output Transformation Server ebXML Messaging - User Guide (VDTOTS-H-UXM)*.

5.3.7.4 EbXmlEvent

The **EbXmlEvent** component connects to a started MshService and waits for messages to arrive. Once messages arrive, a process flow is started. The process flow can then inspect the metadata available in the ebXML message, make decisions, and access payloads.

For more information, see *OpenText Output Transformation Server ebXML Messaging - User Guide (VDTOTS-H-UXM)*

5.3.7.5 FileEvent

The **FileEvent** watches for a specific file type, or a file matching a specific name filter to appear in a designated location (a directory or drive/disk). Once the file appears it can be processed.

This method is well suited to handling large input files. When the file is found in the directory, its file extension is changed, so that it will no longer match the file mask. This is to prevent it from accidentally being picked up again on the next scan. The file extension is changed again when processing is complete, so you know that the file is ready to move to storage or delete. Because the file is being renamed, both read and write access is required.

Sample: SampleFileEvent_Isv2File.xFileEvent

FileEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for file events:

DirsToScan	Options to control directory scanning. These values are required.
Dir	Specifies the directory to scan.
Filter	Contains a regular expression file mask, such as `*.csv`, `abc*.*`, `(\d{9}).xml`, etc., which describes the name of the input file that triggers the process.

ExclusionList	Specifies the regular expression used to exclude files from being picked up. Examples include *.xml, MyFil?.xml, ***.xml. Two or more regular expressions must be delimited by '|' (Example: *.xml|*.zip).
NewExtBefore	Indicates the extension that is added to the filename during processing. This is so that Output Transformation Server will not keep processing the same file.
NewExtAfter	Indicates the extension that is added to the filename when processing is complete. This lets you know that the file has been processed. If you need to rerun the test, change the name back to its original extension.
NewExtError	Indicates the extension to add to the filename if the job fails. The default value is **_failed_timestamp_newExtAfter**.
DeleteAfterProcessed	Determines whether or not the file will be deleted after successfully processing the file.
CaseSensitive	Determines whether file names in the scanned directories are case-sensitive.
Sorting	Determines the sorting criteria for the files selected during a scan.
Sort	Indicates whether to sort files alphabetically, by size, or by last modified date. The default value is **NONE**.
Order	Specifies the sort order to apply to the filtered list of files. Select either **ASCENDING** or **DESCENDING**.
SortLevel	Specifies the level of sorting within the file event. When set to **FILEEVENT**, files from all the directories in the file event will be sorted together; with **DIRECTORY**, files will be sorted one directory at a time. This parameter will be ignored if **Sort=NONE**.
Verbose	Designates whether or not to report a scan interval in your debug logs if no files are found for processing. The default setting is false. Unless you have a specific reason to record empty scans in your logs, it is recommended that this parameter remain set to false to reduce clutter in your debug log. Regardless of the parameter's setting, when files are found for processing an INFO level message is always issued showing the number of files located and the time in milliseconds the scan took.

The next four parameters set the name of job variables that will be assigned values, which are all or part of the input file name. These parameters are all optional. These variables can be used later in naming the output file(s).	
FullnameVariable	Indicates the name of the job variable to be set to the full name of the file being processed: `/path/filename.ext` If a variable name is specified as PATH then it can be dereferenced using `{PATH}`.
PathVariable	Indicates the name of the job variable to be set to the path (directory) of the file being processed: `/path`
FilenameVariable	Indicates the name of the job variable to be set to the filename of the file being processed. If a variable name is specified as FILE, then it can be dereferenced using `{FILE}`.
ExtensionVariable	Indicates the name of the job variable to be set to the extension of the file being processed: `.ext`. If a variable name is specified as EXT, then it can be dereferenced using `{EXT}`.
AppendUniqueId	Specifies whether or not to append a unique ID to the end of the file name.
BatchOptions	Options for batching files. All files from the same batch will be passed to the process flow as a multiXDoc and processed in one job.
BatchSize	Specifies the number of files to scan to create one batch. The default value is 1.
WaitToFill	Determines whether or not to wait for an incomplete batch to reach the defined batch size with more directory scans. By default, this parameter is set to false.
WaitToFillTotalScan	Indicates the total number of directory scans to perform to fill up an incomplete batch. The default value is 1.

5.3.7.6 FileEventMT

The FileEventMT is a multi-threaded version of the FileEvent component, providing enhanced file polling capabilities by using one or more threads to scan directories and store those files in queue. Job submission is also performed on a separate thread pulling from the queue. This ensures that there are always files available to process.

For more information, see "FileEvent" on page 166.

5.3.7.6.1 FileEventMT Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for the FileEventMT component:

<DirsToScan>	Options to control directory scanning. These values are required.
<Dir>	Specifies the directory to scan.
<Filter>	Contains a regular expression file mask, such as `*.csv`, `abc*.*`, `(\d{9}).xml`, etc., which describes the name of the input file that triggers the process.
<ExclusionList>	Specifies the regular expression used to exclude files from being picked up. Examples include *.xml, MyFil?.xml, ***.xml. Two or more regular expressions must be delimited by '|' (Example: *.xml|*.zip).
<NewExtBefore>	Indicates the extension that is added to the filename during processing. This is so that Output Transformation Server will not keep processing the same file.
<NewExtAfter>	Indicates the extension that is added to the filename when processing is complete. This lets you know that the file has been processed. If you need to rerun the test, change the name back to its original extension.
<NewExtError>	Indicates the extension to add to the filename if the job fails. The default value is **_failed_timestamp_newExtAfter**.
<DeleteAfterProcessed>	Determines whether or not the file will be deleted after successfully processing the file.
<CaseSensitive>	Determines whether file names in the scanned directories are case-sensitive.
<MultiThreadedScan>	Select this check box to perform multi-threaded scans (in other words, a thread for each individual directory). If this option is not selected, then a single thread will be used to scan all directories.
<DirectoryScanner>	Specifies what implementation of directory scanner to use.

<ScannerImplementation>	Indicates the scanner implementation used by this file event. Select **Ant** to use the default implementation, or **Custom** to implement your own custom scanner class as defined by the **ImplementationClass** parameter below.
<ImplementationClass>	Specifies the fully qualified class name of the custom directory scanner implementation to use when **ScannerImplementation** is set to **Custom**.
<Queue>	Specifies sorting parameters for the directory scanner queue.
<IsSorted>	If selected, files will be sorted by the last modified date.
<SortInterval>	Indicates the time between sorts in milliseconds. Sorting will occur either at this time interval, or when the queue exceeds the defined **SortSize** value, whichever occurs first. The default value is 5000 milliseconds.
<SortSize>	Specifies the number of accumulated files that will initiate the sorting process. Sorting will occur when the **SortSize** value is exceeded, or at the specified **SortInterval** value, whichever occurs first. The default value is 100.
<StatusInterval>	Specifies the interval, in milliseconds, at which queue status is logged.
<Verbose>	Designates whether or not to report a scan interval in your debug logs if no files are found for processing. The default setting is false. Unless you have a specific reason to record empty scans in your logs, it is recommended that this parameter remain set to false to reduce clutter in your debug log. Regardless of the parameter's setting, when files are found for processing an INFO level message is always issued showing the number of files located and the time in milliseconds the scan took.
The next four parameters set the name of job variables that will be assigned values, which are all or part of the input file name. These parameters are all optional. These variables can be used later in naming the output file(s).	
<FullnameVariable>	Indicates the name of the job variable to be set to the full name of the file being processed: `/path/filename.ext` If a variable name is specified as PATH then it can be dereferenced using `{PATH}`.

<PathVariable>	Indicates the name of the job variable to be set to the path (directory) of the file being processed: `/path`
<FilenameVariable>	Indicates the name of the job variable to be set to the filename of the file being processed. If a variable name is specified as FILE, then it can be dereferenced using `{FILE}`.
<ExtensionVariable>	Indicates the name of the job variable to be set to the extension of the file being processed: `.ext`. If a variable name is specified as EXT, then it can be dereferenced using `{EXT}`.
<AppendUniqueId>	Specifies whether or not to append a unique ID to the end of the file name.

5.3.7.7 FtpEvent

The **FtpEvent** is very similar to a file event, but instead of watching for files to be written to a local or networked directory, it watches for files to be uploaded to an FTP site. In order to do this, an FTP user account and password should be set up just for OpenText Output Transformation Server to use.

This method is well suited to handling large input files. When a file is found on the FTP site, its file extension is changed so that it will no longer match the file mask. This prevents it from being accidentally picked up again on the next scan. The file extension is changed again when processing is complete, so you know that the file is ready to move to storage or delete. Because the file is being renamed, both read and write access is required.

The input file is not automatically deleted after it has been processed. That way, if you need to rerun a process that failed, all you need to do is rename the file back to the original name.

Sample: SampleFtpEvent_Isv2File-Ftp.xFtpEvent

5.3.7.7.1 FtpEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for FTP events:

<Port>	Specifies the port to listen on (required).
<Host>	Specifies FTP site to connect to, such as ftp.yourcompany.com (required).
<User>	Indicates the name of user with full permission to access the FTP site (required).

<Password>	Specifies the password for the user listed above (required).
<DirsToScan>	These are required:
<Dir>	Indicates the directory to scan. Leave blank for FTP root. By default, all subdirectories will be also be scanned for files. See Ignoring subdirectory triggers below for strategies in filtering results to a specific directory only. **Note:** Any subdirectories that the FTP user does not have permissions to do an "ls" on will just result in a INFO message in the log.

<Filter>	Contains a regular expression file mask, such as *.csv, abc*.*, (\d{9}).xml, etc. that describes the file name which will trigger the process flow. You can try different regular expressions to do different things. **Ignoring subdirectory triggers**: By default, all subdirectories will be scanned for files of the specified type(s). To only have events from the immediate directory trigger the process flow, you can enter a regular expression filter that causes subdirectory trigger files to be ignored, and/or you can configure several directories to scan. To filter all subdirectory files, enter a regular expression that excludes all subdirectory files. For example, for the directory (**Dir**) '/' the regex ^((/[^/]+)\.xml)$ will trigger a process flow for all xml documents in the main directory '/' but not for any files in its subdirectories. **Note:** A directory listing of every directory and subdirectory under Dir(ToScan) is still performed but the only results triggering FTP events will be the ones in the immediate directory. You can also change the **Dir(ToScan)** to different subdirectories. When changing the Dir(ToScan), whatever is directly under that directory is treated as if it were under "/". For example, setting Dir(ToScan) to "/sub_1/sub_2/" would make that the "root" directory so the given regex would be applied to a filename like "/input.xml" and NOT "/sub_1/sub_2/input.xml". However, be aware that "/sub_1/sub_2/sub_3/input.xml" will still trigger the event since it is an xml file within a subdirectory of the root directory specified. Using the FTPevent Properties or Wizard, you can configure multiple directories to scan, specifying, if you like, directories which contain only files and no subdirectories. This allows you to use more basic regular expressions, while giving you similar results.
<IsBinary>	Specifies whether file should be downloaded as Binary (or ASCII text).
<NewExtDownloading>	If present, specifies that the file will be renamed before downloading.

<NewExtBefore>	Indicates the extension that is added to the filename during processing. This is so that Output Transformation Server will not keep processing the same file.
<NewExtAfter>	Indicates the extension that is added to the filename when processing is complete. This lets you know that the file has been processed. If you need to rerun the test, change the name back to its original extension.
<DeleteAfterProcessed>	Deletes the file once it has been successfully processed.
<RetryDownloading>	Automatically retries the job if the download fails.
<Timeout>	Indicates the number of milliseconds before timeout (per scan).
<PassiveMode>	Specifies whether or not passive FTP mode should be used. This is usually the preferred FTP mode on a firewalled site. Turn off this option if your system only supports active mode.
<SecureMode>	Specifies whether SSH or SSL security is used (or none).
The next 3 parameters set the name of job variables that will be assigned values which are all or part of the input file name (optional). These variables can be used later in naming the output file(s):	
<DirectoryVariable>	Indicates the variable to contain the input file directory.
<FilenameVariable>	Indicates the variable to contain input file name (only).
<ExtensionVariable>	Indicates the variable to contain original input file.
<PersistentStorage Info>	Keeps track of files that have been downloaded and processed so they are not processed again. This is useful when there is no write access for the FTP account.
<PersistentStorage Name>	Indicates a unique name that identifies this PersistentStorage.
<Enabled>	Specifies whether or not to use PersistentStorage.
<Url>	Specifies the JDBC connect string.
<UserName>	Specifies the user name for the underlying database.

<Password>	Specifies the password for the underlying database.
<JdbcDriver>	Specifies the fully qualified class name for the JDBC driver.
<PropertyFile>	Indicates the location for the property file to override default PersistentStorage options. This is an optional parameter.
<SqlDialect>	Specifies the proper SQL dialect to use with the underlying database.

5.3.7.8 HttpEvent

The **HttpEvent** allows processes to be invoked using web services. It opens a port and waits for firewall-friendly HTTP requests to process.

Because the data to be processed is being streamed into memory through a web service, this method works best with smaller files that can be held completely in memory. Size limitations will vary based on available memory.

Both web applications and web service applications can be deployed into Output Transformation Server. Once an application is deployed, it will be picked up by HTTP server-to-server requests. HTTP Server within Output Transformation Server uses Apache Jetty web server as the HttpEvent implementation.

This method of transfer uses one or more client servlet(s) to handle HTTP requests. These can be created by OpenText Professional Services or by your own staff, then deployed on your server(s).

Sample: SampleHttpEvent_Isv2File-Http.xHttpEvent

5.3.7.8.1 HttpService Properties

The HttpEvent works with an HttpService. The HttpService is the actual web server. To use the HttpEvent, an HttpService must first be configured. There is a sample HttpService located in the samples directory. For more information, see "HttpService" on page 254.

5.3.7.8.2 HttpEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for HTTP events:

<HttpServiceConfig>	Specifies a relative path to the HTTP Service configuration file. Associates this event with an HttpService.
<Path>	Contains the path to RunJob servlet.

<IsSOAP>	Specifies whether or not to return a SOAP response.
<RequestSource>	Indicates that when set to **html**, the HTTP Event will treat the request as an RFC 1867 (http://www.ietf.org/rfc/rfc1867.txt) compliant multipart/form-data stream.
<RequestParam>	Specifies the HTTP request parameters.
<Response>	
<ResponseName>	Indicates the web service event response name.
<ResponseType>	Determines how the web service response name is treated. (values: input/result map, job var or a constant).
<ResponseTimeOut>	Indicates the timeout in milliseconds to retrieve the response from the job ticket.

5.3.7.9 JmsEvent

A **JmsEvent** listens on a specified JMS (Java Messaging Service) queue for a message to appear. When a message appears in the queue, it is processed by the specified process flow. Selectors can be used to filter and select desired messages to process.

The JmsEvent provides for highly reliable message delivery.

Java Messages have three parts: a header, a set of properties, and a payload or body. The header contains fields for routing and identification, and the body contains the application data being sent. Properties can be used to recognize what type of message it is. The JmsEvent can be configured to select only messages with specific properties to process.

Because these messages are streamed directly into memory, the size of the messages that can be processed is limited by the amount of memory that is available.

Your JMS server needs to be up and running, and must have an output queue configured for this to work.

Sample: SampleJmsEvent_Isv2File-Jms.xJmsEvent

5.3.7.9.1 JmsEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for JMS events:

<Name>	Specifies the name of the JMS handler, such as WebSphere Liberty.
<IsBinary>	Specifies whether messages are sent as text or as binary data. Using the Java Naming and Directory Interface (JNDI) allows Output Transformation Server to use the built-in capabilities of Java runtime environments for Java messaging. You can set multiple JNDI Property name-value pairs:
<JndiContext>	Indicates the Java Naming and Directory Interface (JNDI) to use. Using a JNDI allows Output Transformation Server to use the built-in capabilities of Java runtime environments for Java messaging. Multiple JNDI property name-value pairs can be set.
<Property>	
<Name>	Sets the name.
<Value>	Sets the value. In most situations, you will set at least two name value-pairs for JNDI Context properties: `java.naming.provider.url` is normally set to `iiop://localhost:2809` for WebSphere Liberty. `java.naming.factory.initial` is set to the Java class name of the initial context factory to use.
<ConnectionFactory>	The connection factory is an administered object that a client uses to create a connection to the JMS provider. A destination is an administered object that encapsulates the identity of a message destination, which is where messages are delivered and consumed. Set these parameters for the Connection Factory and destinations:
<LookupName>	Specifies the Factory name in the JNDI tree.
<User>	Specifies the login name for the user.
<Password>	Specifies the login password.

<IsQueueConnectionFactory>	Specifies the type of connection factory: Queue or Topic.
<Retries>	Specifies the number of retries to create the JMS connection. The default value of **0** means an unlimited number of retries.
<Destination>	
<LookupName>	Specifies the destination name in the JNDI tree.
<IsDurable>	Specifies whether the subscriber is durable.
<Selector>	The JMS message selector allows selection of specific messages for processing based on header field and property references. Leave the next two parameters blank if all messages are to be processed:
<Name>	Specifies the name of the property to be checked.
<Value>	Specifies the value of the property on the messages to process.
<JmsProperties>	Maps JMS property values to job variables.
<BufferSize>	Specifies the size of the buffer used to pipe the data from the XDoc to the output file.
<Transacted>	If set to **True**, the transaction will be rolled back and the message will be put back on the queue/topic.

5.3.7.10 MailEvent

The **MailEvent** watches an email account and then processes the attachments received in incoming email messages. It supports both POP3 and IMAP mail protocols.

In order to use mail events, an email account needs to be set up for OpenText Output Transformation Server to use.

This type of event is better for smaller files, since the mail server might be configured to reject large attachments. email also does not transfer files as quickly as some of the other methods.

Sample: SampleMailEvent_Isv2Filemail.xMailEvent

5.3.7.10.1 MailEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for mail events:

- **Protocol**. Specifies the email protocol used by your mail server.
- **Port**. Designates the port to listen in on.
- **Host**. Specifies the name of mail server.
- **User**. Specifies the name of the email user.
- **Password**. Specifies the password for the user named above.
- **Mailbox**. Specifies the mailbox from which to retrieve messages. For POP3 protocols, the mailbox name can only be "Inbox".
- **Timeout**. Specifies the number of milliseconds before timeout (per scan).
- **Auth_mode**. Specifies whether to use standard or secure email access.
- **EmailAddressVariable**. Specifies a job variable that will contain the sender's email address. This can be used later by other processes if you need to send a reply.
- **EmailSubjectVariable**. Specifies the name of the job variable to be used as the sender's email subject.
- **ContentTypeVariable**. Indicates the name of the job variable specifying the content type for the attachment. When IncludeAttachments is set to FIRST or LAST, this parameter will be set as a job variable. When IncludeAttachments is set to ALL, this parameter specifies the properties of the MultiXDoc containing the attachments. The default value is **ContentType**.
- **ContentEncodingVariable**. Indicates the name of the job variable specifying content encoding for the attachment. When IncludeAttachments is set to FIRST or LAST, this parameter will be set as a job variable. When IncludeAttachments is set to ALL, this parameter specifies the properties of the MultiXDoc containing the attachments. The default value is **ContentEncoding**.
- **IncludeAttachments**. Determines whether or not email attachments will be retrieved and added to the input map. Attachments that are retrieved will be included in the output file. Options are:
  - **NONE**. No email attachments will be included in the output file.
  - **FIRST**. Only the first email attachment in the list will be retrieved and included in the output file. Note that the attachment is stored in a single XDoc, which may not show the original attachment name.
  - **LAST**. Only the last email attachment in the list will be retrieved and included in the output file. This is the default value. Note that the attachment is stored in a single XDoc, which may not show the original attachment name.

  - **ALL**. All email attachments in the list will be retrieved and the input map will be populated with a MultiXDoc which lists the attachments by name.

- **MessageBody**. Settings to control whether or not to retrieve and populate the input map with the body of the email. Note that only text-based content can be retrieved. Images and other types of multipart content cannot be extracted.

  - **Type**. Specifies where the body of the message will be populated. Options are:

    - **JobVar**. Populates job variable with message body string.
    - **inputXDoc**. Populates the input map with an XDoc containing the message body.
    - **resultXDoc**. Populates the result map with an XDoc containing the message body.

  - **Key**. Specifies the key to use when populating the message body. When this field is left empty, the message body will not be populated.

5.3.7.11 IBM WebSphere MQEvent

An IBM WebSphere **MQEvent** listens on an MQ Server for a message to appear. Once a message is found and retrieved, a process flow will be executed to process that message.

When the message appears in the queue, it is processed by the specified process flow.

A WebSphere MQEvent:

- Watches for data to appear in a WebSphere MQ queue to process.
- Uses Selectors to filter and select desired messages to process.
- Provides highly reliable and cost effective delivery.
- Limits data size by memory capacity.
- Requires WebSphere MQ Server to be up and running.
- Needs a WebSphere MQ license deployed to Output Transformation Server. For details on deploying a license, see *OpenText Output Transformation Designer - User Guide (VDTOTS-H-UTD)*.
- Requires that a *queue connection* needs the queue to be configured, while a *topic connection* needs the topic to be set up in advance.

WebSphere MQ messages have three parts: a header, a set of properties, and a payload or body. The header contains fields for routing and identification, and the body contains the application data being sent. Properties can be used to recognize what type of message it is. The WebSphere MQEvent can be configured to select only messages with specific properties to process.

Sample: SampleMqEvent_Isv2File-Mq.xMqEvent

5.3.7.11.1 IBM WebSphere MQEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for WebSphere MQ events:

<Name>	Specifies the name of the WebSphere MQ handler.
<IsBinary>	Checks whether or not messages should be sent in binary mode, rather than text mode.
<ServerConnection>	Gives details for the server connection. The MQEvent uses the details from the local queue if this parameter is left empty.
<Host>	Identifies the remote host for the server connection. For local connections, leave this parameter empty.
<Port>	Identifies the remote port for the server connection. The default port is 1414. For local connections, leave this parameter empty, although it may have a value that will be ignored.
<Channel>	Identifies the server connection channel for the server connection. For local connections, leave this parameter empty, although it may have a value that will be ignored.
<MQManager>	Specifies the details for the WebSphere MQ Manager.
<Name>	Specifies the name of the WebSphere MQ Manager.
<User>	Specifies the WebSphere MQ user.
<Password>	Lists the WebSphere MQ user's password.
<IsQueueConnection>	Specifies whether this is a queue or a topic connection.
<Retries>	Provides the number of retries if queue connection fails.
<Destination>	Lists the name of the queue. Contains a set of WebSphere MQ destination objects, such as queues and topics.
<Name>	Identifies the Destination name in WebSphere MQ.
<IsDurable>	Checks if there is a durable subscriber.

<Selector>	Specifies the name and value of the selector to filter incoming messages.
<Name>	Identifies the name of the selector.
<Selector>	Specifies the value for the selector.
<MqProperties>	Maps the value of a job variable to a WebSphere MQ property.
<Property>	States the name of the WebSphere MQ property.
<JobVar>	Identifies the name of the job variable that will store the WebSphere MQ property value.
<BufferSize>	Specifies the size (in bytes) of the reused buffer to pipe the data from the XDoc to the output file.
<Transacted>	If true, the transaction will be rolled back and the message will be put back on the queue or topic connection.

5.3.7.12 MSMQEvent

MSMQ is a Microsoft message queue utility. The **MSMQEvent** watches for a file in the Microsoft message queue and then starts the related process flow.

For information about initializing and preparing MSMQ to be used with Output Transformation Server, see "Overview of MSMQ Connectors" on page 156.

5.3.7.12.1 Configuring an MSMQEvent

The MSMQEvent watches for a file in the Microsoft message queue and then starts the related process flow.

Create a new MSMQEvent by choosing the component from the Components list. Once it is in the development window, double-click it to open the properties form and set the configuration parameters.

This component includes a wizard for easy setup, or changes can be made directly in the Properties form.

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

Enter the following values for the given parameters:

<Name>	The name entered must be the same as the MSMQ message queue this event will watch for files from.
<DllQueuePath>	Enter the full Windows path to the OpenText-supplied `MsmqJava.dll` file. If this path is left blank, Output Transformation Server will look for the DLL in the Java library path (`C:\Java\libs\`). The installer places the file in the `<install_home>\lib\dll` folder.
<IsBinary>	Select if messages have to be uploaded in binary mode. Clear if messages are to be uploaded as text.
<MqHost>	Enter the computer name or IP address of the machine where the MSMQ message queue is configured. If this is the same machine as the Output Transformation Server application, enter localhost.
<Action>	Message processing can be done in one of two ways:
<SaveToFile>	Choose this option to save each received message before it is sent to the process flow and then delete the message once the process flow has completed successfully. If the process flow does not finish successfully, the message is retained. This type of processing is time and resource intensive, but useful during testing.
<OnFly>	Choosing the this action is quicker and less resource intensive because files are passed through to the process flow without being saved. If an exception occurs during the process flow, the message is then saved to file.
<ActionPath>	Enter the path to where messages will be saved.
	Note: You must enter a file path whether the Action chosen was SaveToFile or OnFly because if the process flow fails, the message being processed needs to be saved. If there is no folder path, the message cannot be saved.
<Destination>	This section is used to define the target queue for receiving messages.

<QueueTopic>	Enter the type of queue. This will be private or public (see the MSMQ documentation), unless you have defined a custom queue type. **Note:** If you have created a customized URL for the queue connection, enter the URL here and do not enter anything at Name (below) or MQHost (above). If a custom URL is entered here, it will override anything entered in Name or MQHost. The URL would be in the format: `url= Direct=OS:machinename\ \private$\\queue` or `url= Direct=TCP:121.0.0.1\ \private$\\queue`
<Name>	Enter the exact name of the queue.
<Timeout>	Enter the number of milliseconds between calls to the MSMQ to check for waiting messages. Zero (0) means messages are read and processed immediately.
<Selector>	Use the selector options to filter which messages will be passed to the process flow. If no filtering is required and all messages will pass to the process flow, do not add selector criteria. Filtering can be implemented by message label or correlation id. Enter a string or a regular expression at **Label** or **CorrelationID**.

5.3.7.12.2 MSMQ Job Variables

Once the message is received by the MSMQEvent, the following variables will be populated and available for use in the process flow that follows the event:

Variable name	Value
Label	The MSMQ message label.
CorrelationID	The MSMQ message correlation ID.

5.3.7.13 SocketEvent

The **SocketEvent** watches a specified port and processes incoming data. It provides fast and reliable handling of all file sizes and multiple chunks of data can be delimited by either a header or terminator character.

Data is not encrypted.

Sample: SampleSocketEvent_Isv2File-Socket.xSocketEvent

5.3.7.13.1 SocketEvent Properties

Note: This component also includes standard event parameters and clustering parameters. For more information, see "Standard Event Parameters" on page 164 and "Clustering Parameters" on page 165.

The following parameters are set for Socket events:

Port	Specifies the number of the port to check for data.
Timeout	Specifies how long to wait for data before timing out.
MaxConnection	Specifies how many client connections are allowed.
PollingInterval	Specifies the number of milliseconds before idle socket server connection is closed.
Bind	Specifies the IP address to bind the socket to.
IsBinary	Specifies whether data is treated as a text file or as a binary object.
Header	Specifies the length of the message header.
Terminator	Tells the program how to recognize the end of a message (in hex characters), such as 0x1c;0x0d.
MaxMessageSize	Specifies how big, in KB, a message can be. Set to 0 if there is no size limit.
HandlerInputBuffer	Specifies the input buffer size for the handler.
HandlerOutputBuffer	Specifies the output buffer size for the handler.
KeepClientSocket	Specifies whether the client socket is kept open.

HandlerPool	A socket handler reads, decodes, processes, and replies to incoming messages. You can have a pool of handlers all doing the same thing, to keep up with many incoming messages. The following parameters are set for the Socket event handler pool:
MaxActive	Specifies how many handlers can be active.
MaxIdle	Specifies how many handlers can be idle.
MinIdle	Specifies how many available handlers can still be available when a new one needs to be made.
MaxWait	Specifies how long to wait for objects before generating and exception, if all handlers are busy.
MinEvictableIdleTimeMillis	Specifies how long to let a handler sit idle waiting for work before shutting it down.
NumTestsPerEvictionRun	Specifies the default number of objects to examine per run in the idle object evictor.
TestOnBorrow	If selected checked, objects will be validated before borrowing the object. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another.
TestOnReturn	If checked, objects will be validated before being returned to the pool.
TestWhileIdle	If checked, objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool.
TimeBetweenEvictionRunMillis	Indicates how often to check for idle handlers (in milliseconds).
WhenExhaustedAction	Specifies the action to take when an object is borrowed and when the pool is exhausted (the maximum number of "active" objects has been reached). Values: 0, 1, or 2.
Value: 0	When the maximum number of active objects has been reached, the borrowObject() method should fail.
Value: 1	When the maximum number of active objects has been reached, the borrowObject() method should block until a new object is available, or the maximum wait time has been reached.
Value: 2	When the maximum number of active objects has been reached, the borrowObject() method should simply create a new socket event handler.

HostVariable	Specifies the variable that will contain the host name, for use in associated processes.
PortVariable	Specifies the variable that will contain the port number, for use in associated processes.
Acknowledgement	The following parameters define how the socket acknowledgement is performed:
Enabled	If enabled, an acknowledgement will be sent out using the same socket connection the original message was received on.
IgnoreExceptions	If enabled, exceptions in the process flow triggered by this event will not be sent back as part of the acknowledgement.
ResponseName	Specifies the name that will be used with the ResponseType, once the appropriate name is created and filled within the resultMap, inputMap, or jobVar - the response will be sent back.
ResponseType	Determines how the response specified in ResponseName is used.
Result_Map	Looks for the response in the resultMap.
Input_Map	Looks for the response in the inputMap.
JobVar	Looks for the response in a job variable.
Constant	The contents of this field will be sent back as the response.
ResponseTimeOut	Specifies the time in milliseconds before the acknowledgement response will time out.

5.4 Processes

A **process** is a Output Transformation Server component that can be used in a process flow. A process has a single input and one or many outputs. Processes perform some task and then determine which output to go to next.

For more information on process flows, see “Process Flows“ on page 63

5.4.1 Compression Processes

The topics in this section cover the processes you can use to compress and decompress data.

5.4.1.1 CompressDataProcess

The **CompressDataProcess** component compresses the size of the information in order to take up less memory and bandwidth resources.

CompressDataProcess Properties

CompressDataProcess has the following parameters:

<Format>	Specifies the compression format to use. You can select between **ZIP** and **GZIP**.
CompressionLevel	Indicates the level of compression to use. You can select from **NO_COMPRESSION**, **DEFAULT_COMPRESSION** (normal compression), **BEST_SPEED** (compresses at the fastest speed but at a lower compression level), and **BEST_COMPRESSION** (compresses to the smallest size possible but at a slower speed). This parameter is applicable only when the **ZIP** format is used.
MultipleEntries	When set to true, each time the process is executed in a loop, a new entry will be added to the existing XDoc. When set to false, a new XDoc will be created for each entry.
UseRelativePath	When set to true, the relative path for the **ZipEntryName** variable will be used instead of the canonical file path. By default, this parameter is set to false.

5.4.1.2 CompressDirectoryProcess

The **CompressDirectoryProcess** component compresses an entire directory in order to preserve the directory structure, take up less memory and reduce bandwidth resources.

CompressDirectoryProcess Properties

The CompressDirectoryProcess component has the following parameters:

SourceDir	Indicates the source directory to compress. Enter the file path for the directory you want to compress, or click the **Ellipsis** button to browse to the desired directory.

Includes	Specifies the regular expression for the files to compress from the source directory. Use the editor to enter the desired expression. For examples of valid entries, see "FileCopy" on page 192.
Excludes	Specifies the regular expression for the files to ignore during the compression process. Use the editor to enter the desired expression. For examples of valid entries, see "FileCopy" on page 192.
CompressionLevel	Indicates the level of compression to use. You can select from **NO_COMPRESSION**, **DEFAULT_COMPRESSION** (normal compression), **BEST_SPEED** (compresses at the fastest speed but at a lower compression level), and **BEST_COMPRESSION** (compresses to the smallest size possible but at a slower speed). This parameter is applicable only when the **ZIP** format is used.
SpanFiles	Splits the directory into multiple compressed files.
Enabled	If selected, the directory will be split into multiple compressed files when the **Threshold** value is exceeded.
Threshold	Specifies the maximum size of the compressed file. When this value is exceeded, the directory will be split into multiple compressed files. For **Size**, enter the size of the compressed file. The default value is 650. For **Unit**, select either kilobytes, megabytes or gigabytes to specify the unit to use when calculating compressed file size.
WriteToFile	Select the check box to write to a physical file.
FileName	Enter the path and name of the physical file. If the directory spans into multiple files, the index will be added to the file name.

5.4.1.3 DecompressDataProcess

The **DecompressDataProcess** component decompresses compressed data so that it can be used.

DecompressDataProcess Properties

The DecompressDataProcess component has the following parameters:

Format	Specifies the decompression format for the files. You can select between **ZIP** and **GZIP** file formats.

5.4.1.4 DecompressDirectoryProcess

The **DecompressDirectoryProcess** component decompresses a compressed directory and its subdirectories/files to a specified location.

DecompressDirectoryProcess Properties

The DecompressDirectoryProcess component has the following parameter:

DestinationDir	Indicates the destination directory where the directory will be decompressed. Enter the file path or click the **Ellipsis** button to browse to the desired location.

5.4.2 EDI Processes

Topics in this section cover the EDI properties.

5.4.2.1 EDI977AckHandler Process

The **Edi977AckHandler** reads incoming ACK messages, gathers the appropriate message IDs and acknowledgement codes, and passes them to the message service for handling. This process will trigger one of two connected processes:

- If the message ID is not recognized, the **IDNotRecognized** flow will be triggered.
- If the ID is recognized, the **IDRecognized** flow will be passed the message, the ACK, and the previous job vars.

Edi977AckHandler Properties

The Edi977AckHandler has the following parameters:

MessageService	Specifies a pointer to the configuration file for the message service that sent the message for this acknowledgement. This must match the message service set in the EDI Message Sender Process. For more information, see "EdiMessageSender Process" on page 191.
SegmentSeparator	Specifies the character that is used for separating segments.
ElementSeparator	Specifies the character that is used for separating elements.
ComponentSeparator	Specifies the character that is used for separating components.
EscapeCharacter	Specifies the escape character.

5.4.2.2 EdiMessageSender Process

The **EdiMessageSender** reads outgoing messages and passes them to the message service along with a message ID and any job variables created for persistence.

EdiMessageSender Properties

The EdiMessageSender has the following parameters:

MessageServiceInstance	Specifies a pointer to the configuration file for the message service that sent the message for this acknowledgement.
SegmentSeparator	Specifies the character that is used for separating segments.
ElementSeparator	Specifies the character that is used for separating elements.
ComponentSeparator	Specifies the character that is used for separating components.
EscapeCharacter	Specifies the escape character.

5.4.2.3 EDISplitterProcess

The **EDISplitterProcess** splits EDI bundles by type.

EDISplitterProcess Properties

The EDISplitterProcess has the following parameters:

SegmentSeparator	Specifies the character that is used for separating segments.
ElementSeparator	Specifies the character that is used for separating elements.

ComponentSeparator	Specifies the character that is used for separating components.
EscapeCharacter	Specifies the escape character.
CreateAck	Controls whether or not to create an acknowledgement for the incoming.
DirsToScanForDictionaries	You can set multiple sets of DirsToScanForDictionaries parameters.
Dir	Specifies the name of the directory to check for the EDI data dictionaries.
Filter	Specifies the name mask for the data dictionary name.
NewExtBefore	Leave blank.
NewExtAfter	Leave blank.
AckVersionToCreate	Specify 3020, 4010, or SameAsInput to control the format of the Acknowledgement.
RejectDataValidation Errors	Determines if data should be rejected when validation errors occur.

5.4.3 File Processes

Topics in this section describe the available file processes.

5.4.3.1 FileCopy

The **FileCopy** component copies files from one location to another. Using regular expressions, you can also designate that only files found within the given directory matching specific conditions are copied to the destination location.

FileCopy Properties

The FileCopy component has the following parameters:

SourceDir	Indicates the source directory to search for files from.
Includes	Specifies the regular expression for the files to copy from the source directory. Use the editor to key in the expression, based on the examples and example patterns shown below.
Excludes	Specifies the regular expression for the files to ignore during the copy process. Use the editor to key in the expression, based on the examples and example patterns shown below.

DestinationDir	Indicates the destination directory where the files are copied to. If the directory does not exist, it is automatically created.
OverwriteDestinationFiles	If true, the component will overwrite existing files in the destination directory.
ReliableTransfer	This process requires Reliable Transfer functionality. For more information, see "ReliableTransfer" on page 278

Example

In this example, the **FileCopy** component copies the files under the `images` folder, except those files with `.gif` file extensions, to the folder `xenos/lib/images`.

SourceDir = "`/<install_home>/<version>/docs/products/<productname>/UserGuide`"

Includes = "`**/UserGuide/images/*`"

Excludes = "`**/UserGuide/images/*.gif`"

DestinationDir = "`xenos/lib/images/`"

OverwriteDestinationFiles = **Yes**

Example patterns:

Working with the file system `/<install_home>/<version>/docs/products/<productname>/UserGuide/images/`, here are some patterns you can write in the **Includes** or **Excludes** parameters:

Pattern	Examples

`**/docs/*`	Matches all files in the `docs` directories that can be located anywhere in the directory tree.
	
	Matches:
	
	`docs/products/`
	
	`<install_home>/<version>/docs/products/`
	
	But not:
	
	`<version>/docs/outputtransformationserver/UserGuide/products/`
	
	(`outputtransformationserver/UserGuide` part does not match)

Pattern	Examples

`outputtransformationserver/<version>/**`	Matches all files in the `/outputtransformationserver/<version>` directory tree.
	
	Matches:
	
	`outputtransformationserver/<version>/initialFiles/common/logs/readme.txt`
	
	`outputtransformationserver/<version>/docs/`
	
	But not:
	
	`outputtransformationserver/readme.txt`
	
	(`<version>/` part is missing)

`docs/**/UserGuide/*`	Matches all files in the `UserGuide` directories that are located anywhere in the directory tree under `docs`.
	
	Matches:
	
	`docs/products/outputtransformationengine/UserGuide/images/`
	
	`docs/products/outputtransformationserver/UserGuide/images/`
	
	But not:
	
	`docs/products/outputtransformationserver/UserGuide/lib/archived/images/`
	
	(`lib/archived` part does not match)

`**/outputtransformationserver/**`	Matches all files that have an `outputtransformationserver` element in their path, including `outputtransformationserver` as a filename.

5.4.3.2 FileOperationsProcess

The FileOperationsProcess component allows you to execute base file and directory operations, such as copying, moving, renaming, and deleting, as well as creating directories.

FileOperationsProcess Properties

The FileOperationsProcess component has the following parameters:

Operation	Select a file operation type from the dropdown list.
CopyOperationParm	Allows you to copy files or directories to a new location.
Source	Use these parameters to specify the files you wish to copy.
Directory	Indicates the root directory of the file set to copy.
Includes	Specifies the files to be included in a comma- or space-separated list of file patterns. (For examples, see "Include and Exclude Pattern Syntax" on page 197.) If left blank, all files in the specified directory will be included.
Excludes	Specifies the files to be excluded in a comma- or space-separated list of file patterns. (For examples, see "Include and Exclude Pattern Syntax" on page 197.) If left blank, no files will be excluded.
Casesensitive	Determines whether or not include and exclude patterns are case sensitive. This option is enabled by default.
Target	Designates the directory where the files will be copied.
Flatten	Select this option to ignore the directory structure of the source files, and copy all files into the directory specified by the **Target** parameter.
IncludeEmptyDirs	Select this option to copy any empty directories included in the source file set.
Overwrite	Select this option to overwrite existing files even if they are newer than the files being copied.
MoveOperationParm	Allows you to move files or directories to a new location. Note that this operation will remove the specified files/directories from their original location.

Source	Use these parameters to specify the files you wish to move.
Directory	Indicates the root directory of the file set to move.
Includes	Specifies the files to be included in a comma- or space-separated list of file patterns. (For examples, see "Include and Exclude Pattern Syntax" on page 197.) If left blank, all files in the specified directory will be included.
Excludes	Specifies the files to be excluded in a comma- or space-separated list of file patterns. (For examples, see "Include and Exclude Pattern Syntax" on page 197.) If left blank, no files will be excluded.
Casesensitive	Determines whether or not include and exclude patterns are case sensitive. This option is enabled by default.
Target	Designates the directory where the files will be moved. This value is required. If left blank, the source files will be moved to the dev-studio directory (default location is `<install_home>\dev-studio`).
Flatten	Select this option to ignore the directory structure of the source files, and move all files into the directory specified by the **Target** parameter.
IncludeEmptyDirs	Select this option to move any empty directories included in the source file set.
Overwrite	Select this option to overwrite existing files even if they are newer than the files being moved.
RenameOperationParm	Allows you to rename a single file or directory.
Source	Indicates the file or directory to rename.
Target	Specifies the renamed file or directory.
DeleteOperationParm	Allows you to delete the specified files and subdirectories.
Source	Use these parameters to specify the files you wish to delete.
Directory	Indicates the root directory of the file set to delete.

Includes	Specifies the files to be included in a comma- or space-separated list of file patterns. (For examples, see "Include and Exclude Pattern Syntax" on page 197.) If left blank, all files in the specified directory will be included.
Excludes	Specifies the files to be excluded in a comma- or space-separated list of file patterns. (For examples, see "Include and Exclude Pattern Syntax" on page 197.) If left blank, no files will be excluded.
Casesensitive	Determines whether or not include and exclude patterns are case sensitive. This option is enabled by default.
IncludeEmptyDirs	Select this option to delete any empty directories included in the source file set.
MkdirOperationParm	Allows you to create a new directory.
Directory	Specifies the directory to create. Non-existent parent directories will be created when necessary.
ReliableTransfer	Contains the settings for the ReliableTransfer mechanism that is activated when an error occurs during the transmission of data. For more information, see "ReliableTransfer" on page 278.

Include and Exclude Pattern Syntax

The FileOperationsProcess component uses Apache Ant, which requires that you follow the Ant pattern syntax when specifying which files to include or exclude in an operation. Refer to the Apache Ant documentation for more information about Ant pattern syntax.

You can use the `*` wildcard in your include/exclude pattern where:

`*` matches zero or more characters

`**` matches zero or more directories

For example, `**/*.java` matches all `.java` files in any subdirectory.

Other examples:

`includes="src/com/oreilly/util/*.java"` will include all `.java` files in a particular directory.

`includes="src/**/*.java"` will include all `.java` files in all subdirectories of the root folder.

Note: File names listed in the include/exclude pattern cannot contain any spaces.

5.4.3.3 FileProcess

FileProcess writes the output to a local disk file. Optionally you can specify the following:

- Allow it to overwrite any pre-existing file.
- Append a unique date/time ID to the output filename.

FileProcess Properties

The following parameters are available for this process:

FileName	Specifies the fully qualified path and filename of the output file you wish to write.
WriteMode	Specifies the file writing mode. You can select from the following options:
None	Writes to file if it does not already exist. This is the default value.
Append	Adds to the end of an existing file.
Prepend	Inserts at the beginning of an existing file.
Overwrite	Overwrites the file if it already exists.
Append_UniqueID	Writes to file with a unique ID appended to the end of the file name.
BufferSize	Specifies the size of the reused buffer to pipe the data to the output file.
CreateDirectory	Specifies whether or not to create the file's directory if it does not already exist.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.3.4 GetFile

The **GetFile** component loads a file and puts it in an XDoc for use within a process flow.

5.4.3.4.1 GetFile Properties

The GetFile component has the following parameters:

DirName	Indicates the name of the directory to grab a file off the disk and place it in an XDoc for use within the process flow.
FileName	Specifies the regular expression to grab a file off the disk and place it in an XDoc for use within the process flow. If more than one files are returned from the regular expression, an error will be thrown.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.4 Router Processes

The topics in this section describe the available router processes.

5.4.4.1 InvokeProcessByName

The **InvokeProcessByName** component is used in a process flow to run a process or component that is not constant. This is useful when you do not want to set up a router to route to multiple flows, but want to control what is run based on a certain value within a job variable. The InvokeProcessByName component can run any runnable, such as a component, a transformation, or another process flow.

InvokeProcessByName Properties

InvokeProcessByName has the following parameter:

JobVarName	Indicates the name of the job variable specifying the configuration file of the runnable component. Its default value is RUNNABLE_CONFIG.

Example

In the sample process flow below, the **JobVariableSetterProcess** component is used to set the job variables which will then be executed by the **InvokeProcessByName** component. For more information, see "JobVariableSetterProcess" on page 225.

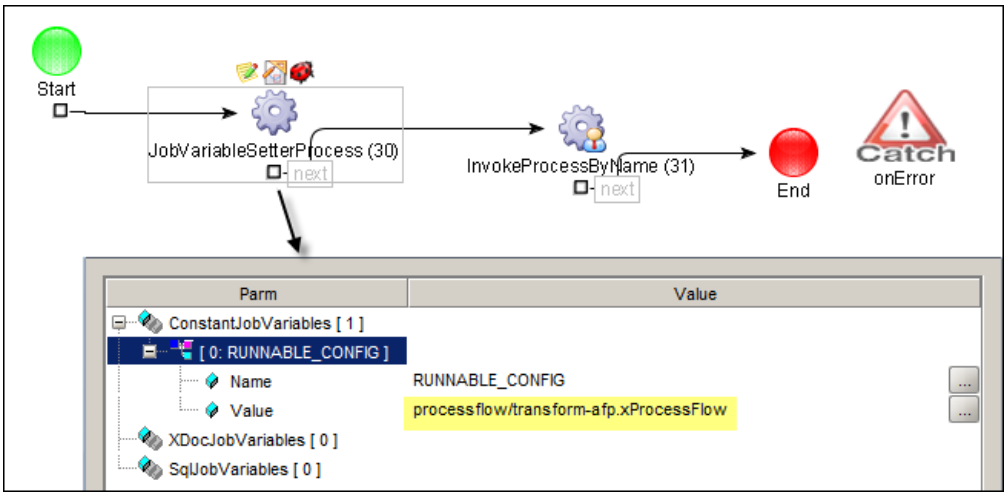


Figure 5-7: JobVariableSetterProcess and InvokeProcessByName in process flow

The **JobVariableSetterProcess** component is configured with the RUNNABLE_CONFIG job variable whose value is set to the location of the process flow to be run within this flow. In this case, `processflow/transform-afp.xProcessFlow` points to another process flow containing an AFP to PDF transformation:

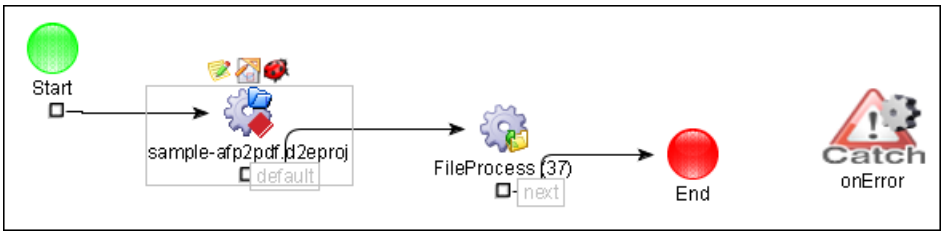


Figure 5-8: Sample AFP to PDF transformation process flow

This AFP transformation flow will be executed by the **InvokeProcessByName** component when the original process flow (at top) is run.

5.4.4.2 JobVariableRouter

The **JobVariableRouter** component is a router that tests user-defined conditions and routes accordingly. The job variable router has dynamic outputs that are generated each time a new condition is added. It is displayed when users select the router process flow from within Output Transformation Server. When a job variable router is initially opened in the Development window, a process flow tab screen opens. The outputs correspond to the conditions from left to right.

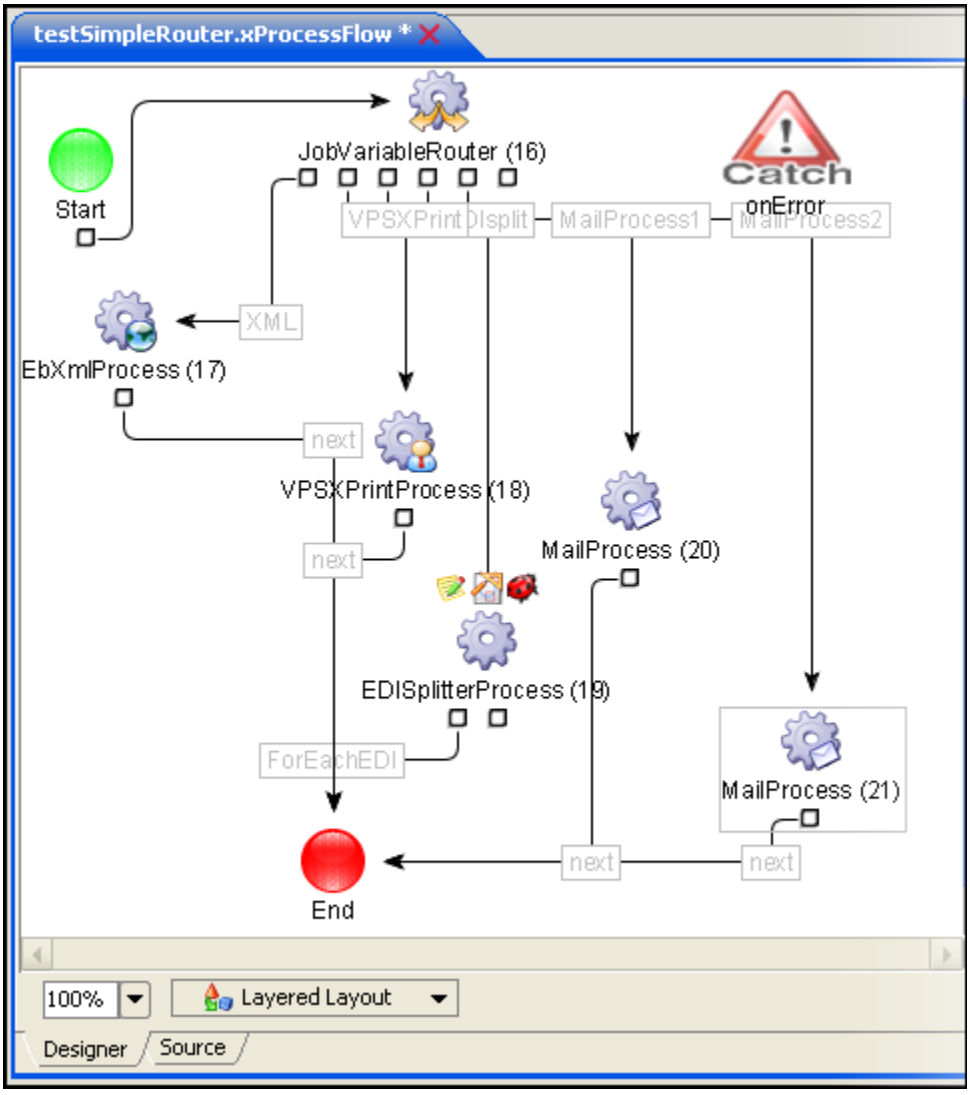


Figure 5-9: Sample JobVariableRouter process flow

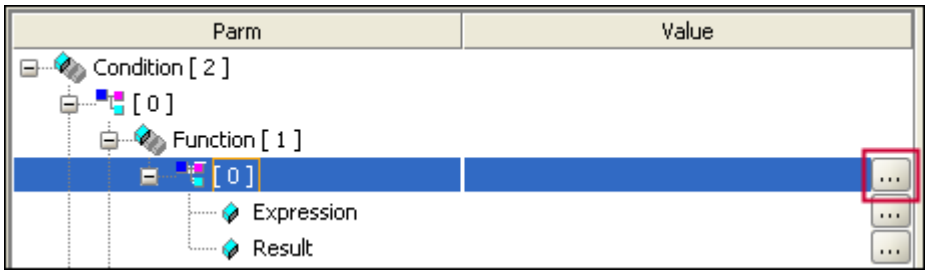
JobVariableRouter Properties

JobVariableRouter has the following properties:

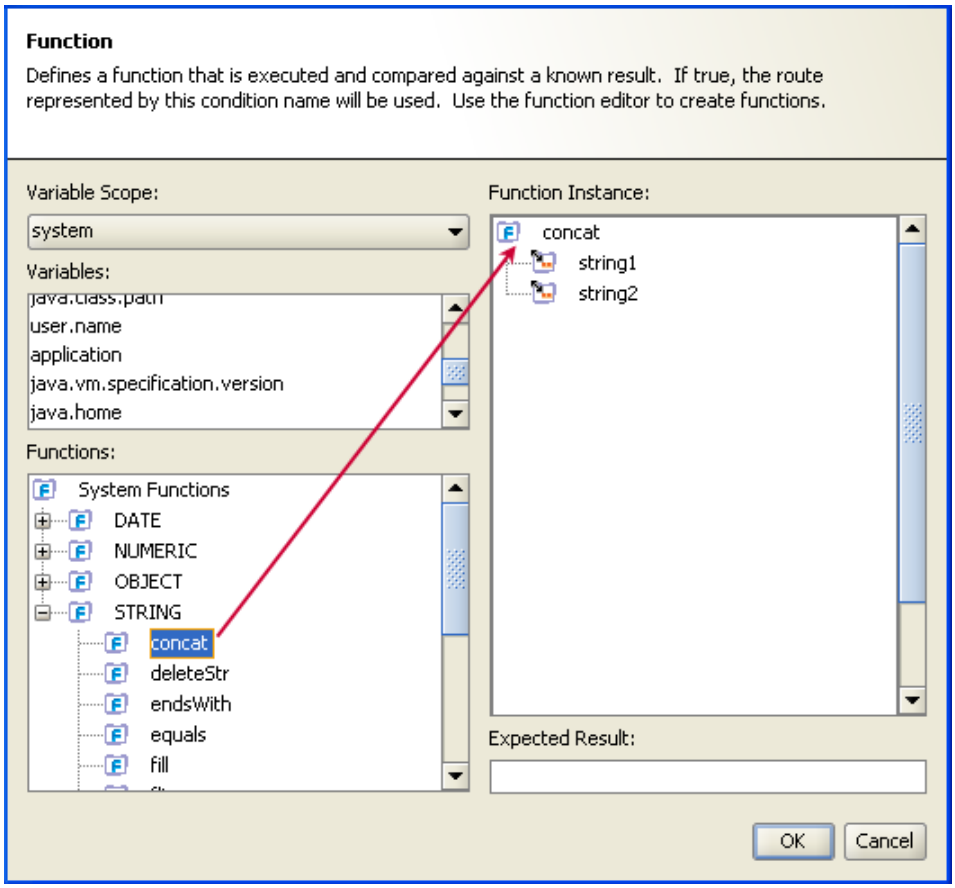
Condition	Designates a list of conditions to compare the values of specific variables against. Each condition corresponds to an output or route. The components of each condition are shown in a tree format. To expand a branch of the tree, click the plus (+) sign. To collapse a branch of the tree, click the minus (-) sign.
Function	Contains a list of functions to determine the results for output. See **Adding a Function** below.
Expression	Specifies a function with arguments. An argument can be another function. This field is populated by the **Function Editor** form, or can be edited directly by clicking the **Ellipsis**,  .
Result	Indicates a value to compare the result of the function execution against. If the condition is met, the **JobVar** setting below is used. This field is populated by the **Function Editor** form, or can be edited directly by clicking the **Ellipsis**,  .
JobVar	Contains a list of job variables and their values, which may include another expression.
Name	Specifies the name of the job variable to set.
ValueType	Indicates the type of job variable. You can select from **LITERAL** or **FUNCTION**.
Value	Specifies the value for the job variable to be set to.
ConditionName	Designates a unique name identifying the condition. It is also used as the output name. If this is the default condition that should be passed through when all other conditions fail, then use default as the condition name.

5.4.4.2.1 Adding a Function

1. Right-click **Function** and choose **Add** to add a new function.
2. Click the top **Ellipsis** button to open the **Function Editor** form.

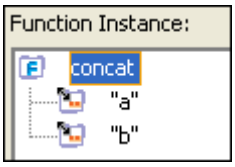


3. Choose your function type from the function palette and drag it to the **Function Instance** box.

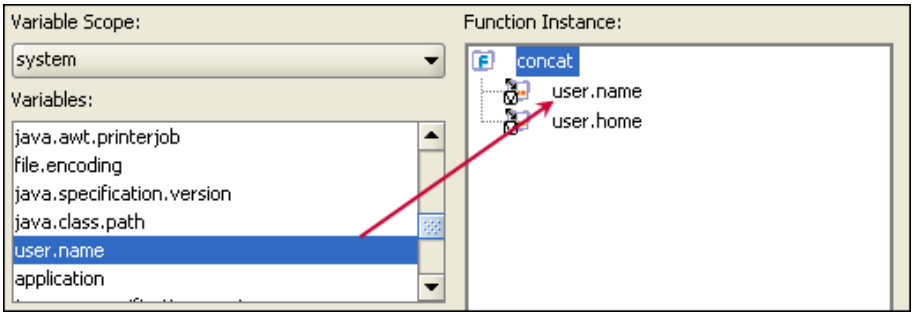


4. Replace the placeholders in the **Function Instance** box with values, variables, or another function.

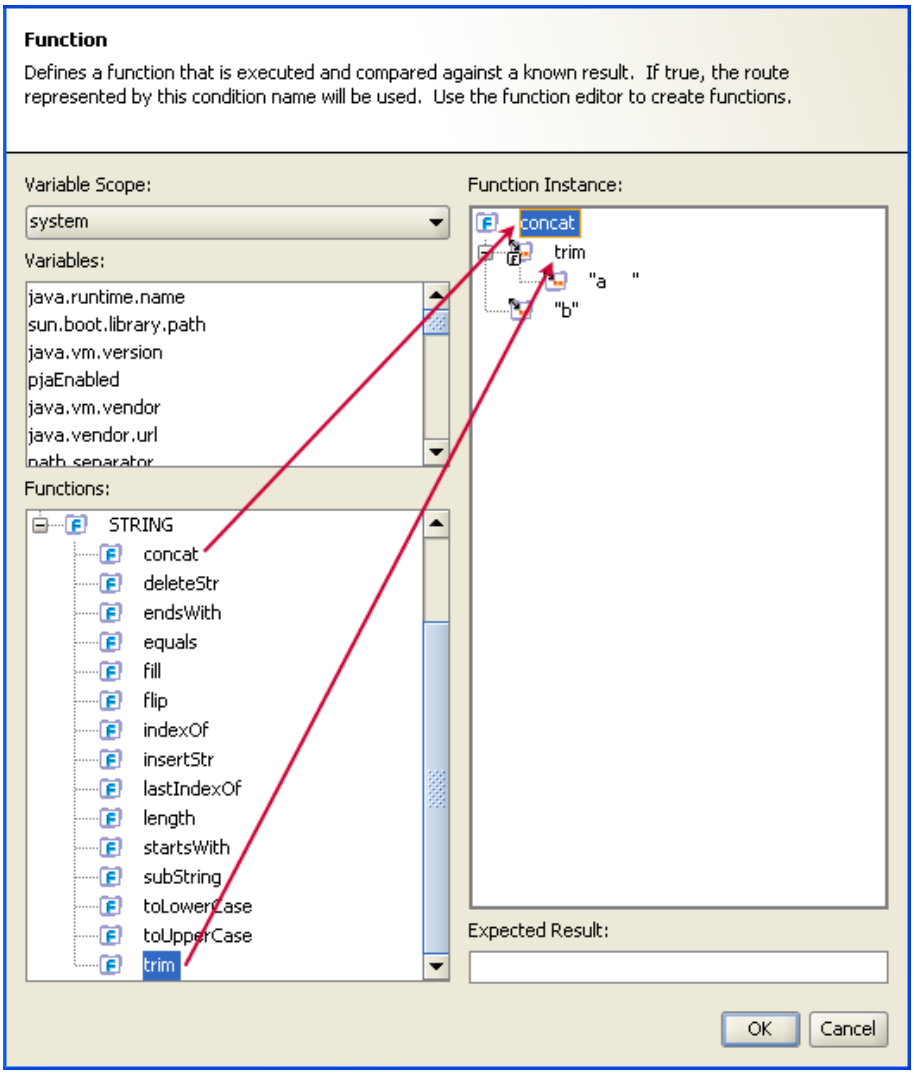
   - For literal values, double-click the placeholder name to make it editable and enclose the literal value in quotation marks.



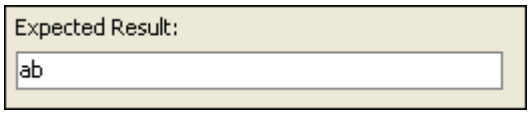
- For variables, drag-and-drop available variables from the **Variables** box onto the placeholders.



- For additional functions, drag-and-drop the function onto the placeholder and then replace this function's placeholders with literal values or variables.

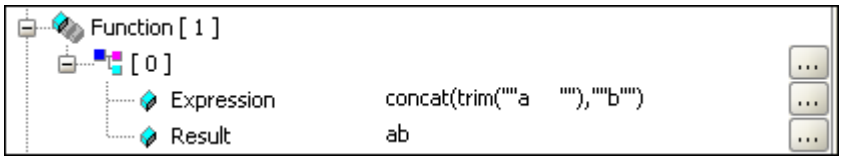


5. Enter the **Expected Result** where indicated. For two of the three above scenarios, the expected result is:



6. Click **OK** to return to the properties.

   The completed function is displayed.



5.4.4.2.2 ContentIdentifier

Job variables are set in the **contentIdentifier** router and are gathered with the job variable router. The content router is found in the Component Template Palette and can be dragged onto the process flow screen. The **contentIdentifier** will route based on content. For more information, see "ContentIdentifier" on page 245.

The **JobVariableRouter** routes traffic through to the appropriate filter depending on the document type. Filter types include CSV, XML, SWIFT and EDI.

5.4.4.3 MultiXDocSplitter

The **MultiXDocSplitter** splits one multiXDoc into individual XDocs. When this component is put into a process flow, a loop will run, repeatedly returning to the multiXDoc to remove one entry each time. This loop will continue until there are no XDocs left.

Two variables will be created from each repeat, which are indexed in JobTicket:

- **xdocEntry**. Indicates the name of the XDoc (if applicable).
- **xdocIndex**. Indicates the index of the XDoc within the multiXDoc list.

In a process flow, any component that produces a multiXDoc (like **Output Transformation Project** or **Data Transformation Project**) would generally precede the **MultiXDocSplitter** component. Extracted XDocs can then continue within a process flow (into a **MailProcess**, for example), or the process can end.

5.4.4.4 TimeRouter

The **TimeRouter** component permits the dispatch of jobs based on a time criteria. The component also allows for multiple ranges to be set up in order to route to different jobs at different times.

The ranges for TimeRouter are dynamic outputs that are generated each time a new condition is added in the Process Flow Editor. The parameters for each range define the start and end points of a time period.

The route to take is chosen based on greater than or equal to the start time and less than the end time. For example, given the following TimeRouter setup:

```
     RouteDef
TimeDef
From
   6:00
To
   20:00
RouteToTake
   A
ConditionName
   A
     RouteDef
TimeDef
From
   20:00
To
```

```
   6:00
RouteToTake
   B
ConditionName
   B
```

The following times will be routed to the specified routes:

- 6:00 - A
- 10:00 - A
- 20:00 - B
- 4:00 - B

TimeRouter Properties

The TimeRouter has the following parameters:

TimeDef	Specifies the times to evaluate for the selected route.
RouteToTake	Indicates the route to take if the current time is within the defined time period.
ConditionName	Specifies a name for the condition.

5.4.4.5 XDocSizeRouter

The **XDocSizeRouter** routes output XDocs in a process flow based on the size of the XDoc. The job variable **xdocSize** is set with the size of the XDoc in the **XDocSizeRouter** process and the XDoc is routed based on this value. The **XDocSizeRouter** does not create or manipulate XDocs, it simply evaluates the size of an XDoc input and outputs it to the appropriate next step in the process flow accordingly.

XDocSizeRouter Properties

The **XDocSizeRouter** has the following parameters:

<Size>	*<Specifies the size of the XDoc file to route.>*
<Units>	*<Indicates the units of the size in bytes, megabytes, or kilobytes.>*

The XDocSizeRouter has three routes:

- **Less.** *<Chooses this route if>* the size of the input XDoc is less than the value set in the parameters.
- **Equal.** *<Chooses this route if>* the size of the input XDoc is equal to the value set in the parameters.
- **Greater.** *<Chooses this route if>* the size of the input XDoc is greater than the value set in the parameters.

When you add an **XDocSizeRouter** component to a process flow, you determine how to route a given XDoc based on its size.

In the following example, if the file is less than 5MB, the output is routed to FileProcessSmall, if it is equal to 5MB, it is routed to FileProcessEqual5MB, and if it is greater than 5MB, it is routed back to the Output Transformation project.

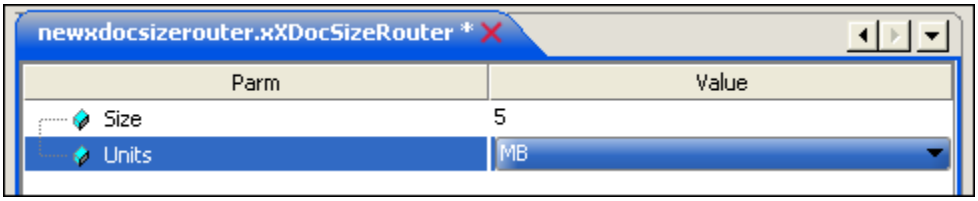


Figure 5-10: XDocSizeRouter properties

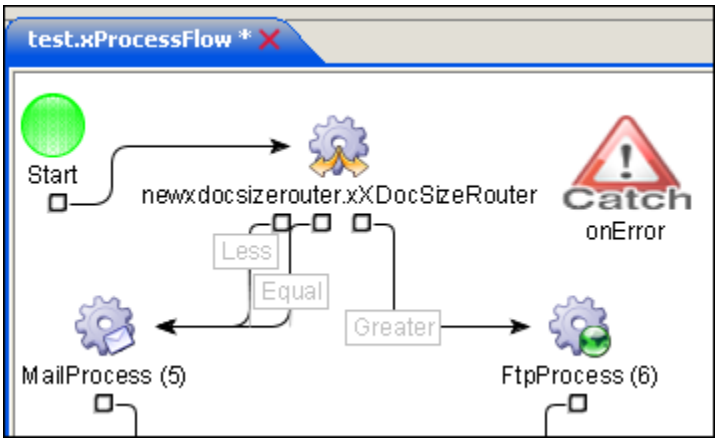


Figure 5-11: XDocSizeRouter in a process flow

5.4.5 Transform Processes

The topics in this section describe the available transform processes.

5.4.5.1 Data Transformation Project

The **Data Transformation Project** component allows you to start and run OpenText Embedded Data Transformation Engine projects within a Output Transformation Server process flow.

Note: You must have Data Transformation Engine licensed and installed alongside Output Transformation Server in order to use this component.

If **Data Transformation Project** is being used with components which produce multiple documents as the expected output, you will require the MultiXDocSplitter component. For more information, see "MultiXDocSplitter" on page 206.

For information about creating and running Data Transformation Engine projects, see *OpenText Embedded Data Transformation Engine - User Guide (VDTOTS-H-UDT)*.

5.4.5.2 Data Transformation Custom Function

When a OpenText Embedded Data Transformation Engine predefined function does not exist to perform the task you need, you can write your own custom function.

For more information, see *OpenText Embedded Data Transformation Engine - User Guide (VDTOTS-H-UDT)*.

5.4.5.3 Output Transformation Project

Adding an **Output Transformation Project** component to a process flow allows you to configure a OpenText Embedded Output Transformation Engine project, which will run within the overall process flow.

If Output Transformation Project is being used with components, which produce multiple documents as the expected output, you may require the MultiXDocSplitter component. This can occur with such Output Transformation Engine components as the DocBreak component.

While Output Transformation Project processes information, it stores in Output Transformation Engine's JobRecord an attribute called the completion code that has two job variables: **d2e.completion.id** and **d2e.completion.desc**. This attribute represents different levels of successful processing and it lives in JobRecord as long as the processing occurs. Other components in the process flow, such as the JobVariablesRouter or a custom component, may need to use the completion code for validation purposes. Once Output Transformation Project ends in the process flow, the completion codes will be set in Output Transformation Server's JobTicket, where downstream components can use the information.

An example of a CompletionCode message in the Job Log appears below:

```
 08:40:07.049-05:00 <TIFF
Generator-writer-0> INFO  : CompletionCode: Value=0,
```

```
 Description=The task has completed successfully without errors
```

Related Links

- “MultiXDocSplitter” on page 206
- “Using Completion Code Information in Process Flows” on page 75
- “JobVariableRouter” on page 201
- “Completion Code Values” on page 76
- *OpenText Embedded Output Transformation Engine - User Guide (VDTOTS-H-UTE)*

5.4.5.3.1 Output Transformation Project Component Properties

The Document Transform component has the following parameters:

WhatToRun	Denotes the configuration file for the Output Transformation Engine project to execute.
FailureMode	Designates the failure mode that defines how the process behaves if any warnings or errors occur during execution. You can choose from **never**, **onWarning**, or **onError**. The never value specifies that the process should not stop while the onWarning and onError values specify that the job should stop upon encountering a warning or error message, respectively. The default value is set to never.
RasterizeAllFonts	Determines whether or not fonts should be rasterized.
CollectIndexerInfoOptions	Determines whether index information from the Index Writer is stored in job variables. This option only applies when there is an Indexer component present in the project.
Enabled	Indicates whether index information from the Index Writer is stored in job variables.
Conditions	Contains the settings related to optional routing based on various conditions.
CompletionCodeConditionParm	Contains parameters related to output conditions based on completion codes.
ConditionName	Specifies a name for the condition.
Operator	Indicates the relational operator to use when comparing the completion code supplied by the Document Transform component at the end of the job.
CompletionCode	Specifies the completion code to compare with.
MessageConditionParm	Contains parameters related to output conditions based on messages.
ConditionName	Specifies a name for the condition.
Filter	Contains settings related to the type of message filter to apply as messages are generated from the executed job.
Enabled	Indicates whether the message filter is enabled.
FilterEntry	Contains settings related to the types of message filters.

MessageType	Identifies the type of messages to filter. You can select from **Any**, **None**, **Debug**, **Info**, **Warning**, **Error**, **Fatal**, **Config**, **Summary**, or **Exception**. The None option effectively disables the filter, which can be useful during testing.
CategoryName	Specifies a regular expression used to filter messages based on the message's original source, such as AfpParser or PdfGenerator. For example, in the sample message header "Info DocumentsCreated(PdfGenerator)", the category name is "PdfGenerator." You can use ".*" to filter all categories.
MessageName	Specifies a regular expression used to filter messages based on the message's name, such as ResNotLoaded. For example, in the sample message header "Info DocumentsCreated(PdfGenerator", the message name is "DocumentsCreated." You can use ".*" to filter all message names.
MessageText	Specifies a regular expressed used to filter messages based on the text within a message. For example, the regular expression ".*Arial-Bold" filters all messages that contain the text "Arial-Bold" anywhere in the message.
FailFast	Designates that the transformation should be aborted as soon as a filtered message condition is met.

5.4.5.3.2 Dynamic Routing with Output Transformation Project Component Output Conditions

The Output Transformation Project component contains functions that permit dynamic routing of transforms, which alters the path of a job within your process flow in response to custom conditions you have configured. There are two types of conditions that can be used to set up this type of adaptive routing: completion codes and messages. Completion code conditions use the standard completion code values produced by the Output Transformation Project component, while message conditions can use a variety of message attributes such as the message's type, original source, name, or even the strings of text within messages. If none of the conditions are met, the job is routed to a default path you define when setting up your process flow.

The output conditions can be customized for a wide range of uses. A sample use case for output conditions is setting up a condition to assist with the generation of accurate accessible PDFs by automatically warning the user of any untagged artifacts. One way a user can achieve this is by creating a message condition that filters out any message text that contains the words "UntaggedContent", which is reported in the logs for all artifacts that remain untagged after processing. By being alerted of the untagged content, the user can go back to correct these errors prior to

continuing with further processing of the job. The conditions can even be configured to abort the transform upon matching the text string in order to ensure that the user can correct any issues before progressing.

For more information on the use of completion codes, see "Using Completion Code Information in Process Flows" on page 75. Additional information about the CompletionCode class can also be found in the Output Transformation Server and Output Transformation Engine Javadocs.

For more information on the log messages that the Document Transform component produces, see *OpenText Embedded Output Transformation Engine - User Guide (VDTOTS-H-UTE)*.

5.4.5.3.2.1 Setting Up an Output Condition

The Create New Output Condition Wizard is available to assist with generating routing conditions for your process flows. (After setting up the output conditions using the wizard, if you want to edit the conditions you must make the amendments manually.

To set up a new process flow with output conditions:

1. In Output Transformation Designer, create a new process flow by opening the New Component Wizard and selecting **Processes** > **Process Flow**.

   After completing the wizard, a tab for your new process flow appears in the Development window.

2. Add the Output Transformation Project component to your process flow. The method for adding the component differs depending on whether you are using an **Existing** Output Transformation Engine project for the component or creating a new project from scratch to be **Shared**.

   The Output Transformation Project component appears in the Development window.

3. Add a process or any other components needed for your project downstream from your Document Transform component to the Development window. Also, configure the parameters for the process or component.

4. Click on the left port below the Output Transformation Project component and draw a connecting line to the process or component that represents the default behavior for output from your Output Transformation Project component. When none of your output conditions are met, the job takes the default route.

   A connection line appears with arrows showing the direction your process flow is going.

   Note: By default, the Document Transform component has only two ports below it for you to draw your connections to other components with. As you make connections between components, more ports for connections to additional components will automatically appear as necessary.

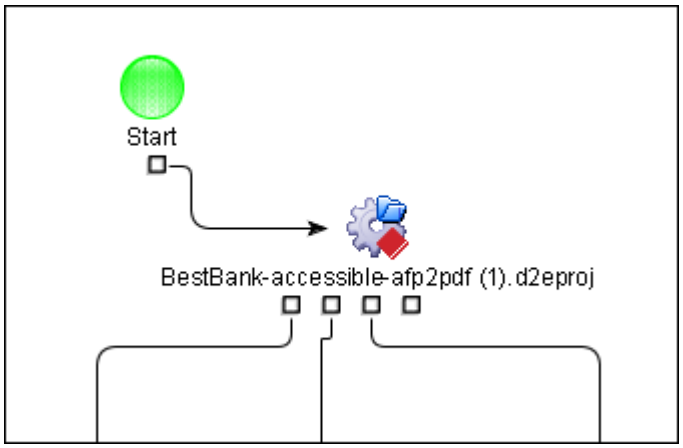


Figure 5-12: Sample Output Transformation Project component with multiple connections to downstream components

5. Next, you are now ready to create an output condition. In the Development window, click on the right-most box under your Document Transform component (if you hover over the box, the **Create New Output** tooltip appears) and draw a connecting line to the process or component you want to set up an output condition for.

   The **Select a Condition Type** screen of the **Create New Output Condition Wizard** appears.

6. On the Select a Condition Type screen, you must select the type of output condition you want to establish. You can choose from **Create completion code based condition** or **Create message filter based condition**. Once you have made your selection, click **Next**.

7. Depending on which type of condition you selected, different screens will appear. If you opted to create a condition based on completion codes, the **Completion Code Condition** screen appears and proceed to step 7a. If you opted to create a condition based on a message filter, the **Message Filter Condition** screen appears and proceed to step 7b.

   The following configurable fields are available on the Completion Code Condition screen:

   - **Condition name.** Specifies a name for the condition.
   - **Operator.** Indicates the relational operator to use when comparing the completion code supplied by the Document Transform component at the end of the job. You must click on the operator symbol to display the dropdown menu in order for you to select a different operator.
   - **Completion code.** Specifies the completion code to compare with.

The following configurable fields are available on the Message Filter Condition screen:

- **Condition name.** Specifies a name for the condition.
- **Abort process when condition is met.** Designates that the transformation should be aborted as soon as a filtered message condition is met.
- **Enable filter.** Indicates whether the message filter is enabled.

8. To create a new message filter condition using the wizard:

   a. On the Message Filter Condition screen, click the **Add** button, +.

      A list of possible messages appear in the table.

   b. You can use the **Category** and **Type** fields to refine the list of messages shown in the table. Adding a regular expression to the **Text** field promotes any matching messages to the top of this list, as well as including the regular expression as part of your filter condition. After selecting a message filter to use, click the **Choose** button, ✓, to confirm your selection.

      The selected message is added to your list of message filter conditions.

   c. In the list of your established message filter conditions, review the message filters and if necessary, click in the cells to perform edits on the conditions. The Category, Name, and Text types can accept the ".*" wildcard to return all messages of the specified type.

9. When you are finished making your selections, click **Finish**.

   Your output condition is saved and the Development window reappears with the Condition name you indicated shown on the connection line between the Document Transform component and your process/component.

10. If necessary, you can add more processes or components in the Development window for your process flow and then repeat steps 5-7 in order to create more output conditions.

    When your process flow is completely configured, save the project and you are ready to run the job.

Related Links

- "Editing an Output Condition" on page 215
- "Creating a Component" on page 84
- "Adding Components to Process Flows" on page 68

5.4.5.3.2.2 Editing an Output Condition

To manually edit an output condition:

1. **Open** your project containing the output conditions you wish to edit in the Development window.
2. Click one of the ports below the Document Transform component.

   The Document Transform component's parameters are displayed in the Properties window.
3. Navigate through the list of parameters to **Conditions** with each grouping of these parameters representing an output condition. You must make any desired changes in the parameter's Value column.
4. When you are finished editing the parameters, **save** your project.

5.4.6 Utility Processes

The topics in this section describe the available utility processes.

5.4.6.1 SystemGcProcess

The **SystemGcProcess** (also known as the Garbage Collection process) runs Java's System.gc() method to release all memory not currently being used by the system. This is especially helpful before or after a memory-intensive process runs to ensure there is enough free memory available.

Garbage Collection can be used at the beginning, middle or end of a process flow.

SystemGcProcess Parameters

This process has the following parameters:

WhenInvoked	Invokes Garbage Collection at a set point in the process flow. Place this process before and/or after a large process to ensure ample memory for the completion of the flow and all other tasks that may be running. The Garbage Collection process will appear in the log as something similar to "Process <SystemGcProcess (5)> has finished. Run time: 172ms."

EndOfJob	Invokes Garbage Collection after all processes in the process flow have run to completion. Add the SystemGcProcess with the EndOfJob setting anywhere in the process flow and it will not start until the process flow has completed. Since this process is actually run by JobManager after job completion, it will appear in the log with a run time of 0ms.

5.4.6.2 XDocBuilder Process

The **XDocBuilder** process constructs an XDoc based on job variables and other XDoc content that are available in the selected job.

XDocBuilder Properties

XDocBuilder has the following parameters:

IgnoreMissingMessageArgs	Determines how to handle missing message arguments. If selected, they are ignored. If not selected, the missed arguments are replaced with place holders, i.e. {0},{1},{2}, etc. In either case, the job continues to completion with no error messages.
MessageArgList	Contains a list of arguments that are used to substitute values into the message format. The position in the collection corresponds to the argument in the message.
Type	Indicates the source where the value will be extracted to use in the message. Acceptable values are **JobVar**, **inputXDoc**, **resultXDoc**, or **Literal**.

Value	Contains a string value whose output depends on the selected type. • If the type is **JobVar**, then the value is the name of the variable. • If the type is **XDoc**, then the value is the name of the XDoc in one of the maps. • If the type is **Literal**, then the value is used directly. This value can be alphanumeric or hex code. Hex code must be entered in the following format as a MessageArgList entry: • Begins with '0x' • Ends with ';' • Cannot be combined with anything other than hex codes. Examples: 0x0a; or 0x0a;0x0d;
MessageFormat	Indicates the message format to use with MessageArgList. For example, 'This {0} is {1} because {2}.'

5.4.7 EbXmlProcess

EbXmlProcess is a runnable component that can be included in a process flow, or run on its own, to send an ebXML message via an MshService.

For more information, see "MshService" on page 261 and *OpenText Output Transformation Server ebXML Messaging - User Guide (VDTOTS-H-UXM)*.

5.4.8 FtpProcess

The **FtpProcess** writes output to a file on an FTP site.

FtpProcess Properties

The following parameters are used for this process:

Port	Specifies the number of the port to be used for the file transfer.
Host	Specifies the name of the FTP host site, such as `ftp.yourcompany.com`
User	Specifies the user name to be used when connecting to FTP.
Password	Specifies the password to be used when connecting.

Timeout	Indicates the amount of time, in milliseconds, to wait before a timeout occurs. The default value is 0.
IsBinary	Designates if the data is to be transferred as a text file or as a binary object.
RemoteDir	Specifies the name of the directory on the FTP site where the file is to be written.
RemoteFile	Specifies the name of the file to be written.
WriteMode	Specifies the file writing mode. You can select from the following options:
None	Checks for an existing file and writes to the file if one is not found. This is the default value.
Overwrite	Overwrites the file if it already exists. No check is done.
Append_UniqueID	Writes to file with a unique ID appended to the end of the file name. No check is done.
SecureMode	Specifies whether you want to use SSL, SSH, or no security when sending the FTP.
ConnectionMode	Specifies whether the local client (PASSIVE) or the server (ACTIVE) initiates the data transfer.
Encoding	Denotes the character coding set to use when issuing FTP commands and transferring files. The default is the UTF-8 character set, which is used by most FTP servers. However, some FTP servers in IBM systems such as z/OS use ISO 8859-1 or another charset.
BatchUpload	Selecting this option will allow you to upload an unlimited number of files to the FTP server in one transaction. If BatchUpload is not enabled, then only one file can be uploaded per transaction.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.9 HttpProcess

The **HttpProcess** sends data to a web server. Parameters set the URL, connection method, and properties.

HttpProcess Properties

The parameters for the HttpProcess are as follows:

BufferSize	Specifies the size of the reused buffer to pipe the data to the output file.
URL	Specifies the URL address of the HTTP server.
MethodName	Specifies the request method name, such as **POST** or **GET**. The default is **POST**. When using **GET**, the only parameters supported are those in the URL.
Acknowledgement	
AckRequired	Indicates whether or not to wait for an acknowledgement from the server. If selected, you can define post request property pairs. POST is the MethodName parameter and the PostRequestProperty parameter needs to be set.
Timeout	Indicates the length of time, in milliseconds, to wait for an acknowledgement before timing out.
ResponseCodeJobVar	Indicates the job variable that stores the HTTP response code.
Proxy	
Proxy Type	Specifies the type of proxy protocol. You can select from **NONE**, **HTTP**, and **SOCKS**.
Host	Indicates the host name or IP address for the proxy server a client wants to connect to.
Port	Indicates the port for the proxy server a client wants to connect to.
User	Specifies the user name to access a proxy server.
Password	Specifies the password to access a proxy server.
HttpAuth	

AuthType	Specifies the type of HTTP authentication or authorization to be used. You can select from **NONE**, **BASIC**, **ENCODEDURL** and **PLAINURL**.
SslEnabled	Indicates whether or not to use Secure Sockets Layer (SSL).
InstallServerCert	Indicates whether the server certificate is auto-installed into the keystore during the SSL handshake. You can select from **No**, **YesAndOverwrite** and **DoNotOverwrite**.
UserName	Specifies the user name for basic HTTP authentication.
Password	Specifies the password for basic HTTP authentication.
TrustStore	
File	Specifies the keystore URL.
Password	Specifies the keystore password.
KeyPassword	Specifies the password that is required to access the private key associated with an alias name. It can be the same as the keystore password.
KeyStore	
HostNameVerification	Indicates whether to perform host name verification during the SSL handshake. The verification is to ensure the host name in the URL that you connect with matches the host name in the digital certificate that the server sends back as part of the SSL connection.
File	Specifies the keystore URL.
Password	Specifies the keystore password.
KeyPassword	Specifies the password that is required to access the private key associated with an alias name. It can be the same as the keystore password.
RequestProperty	Allows you to define multiple property value pairs with either **GET** or **POST** as the MethodName parameter. If the **AckRequired** check box is not selected, the RequestProperty key/value is set. HttpProcess disregards the key/value set for PostRequestProperty.
PropertyName	Specifies the name of the request property.
PropertyValue	Specifies the value of the request property named above.

PostRequestProperty	Allows you to define multiple post request property value pairs, if **POST** is the MethodName parameter and if the **AckRequired** check box is selected. The PostRequestProperty key / value is set, HttpProcess disregards the key/value set for RequestProperty. The key/value are passed to HttpEvent.
PropertyName	Specifies the name of the request property.
PropertyValue	Specifies the value of the request property named above.
ContentDispositionName	Specifies the property name when submitting an XDoc as a stream to an HTTP service. This value, along with the value for **ContentDispositionFileName**, appears in the HTTP header and helps to identify the stream.
ContentDispositionFileName	Specifies the property file name when submitting an XDoc as a stream to an HTTP service. This value, along with the value for **ContentDispositionName**, appears in the HTTP header and helps to identify the stream.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.10 JavaScriptProcess

The JavaScriptProcess component provides users with a means of including scripts for interacting with a running process. API users can employ the component to add custom logic to their processes using JavaScript instead of writing, compiling, and managing custom Java-based components.

The JavaScriptProcess is available in Output Transformation Server's standard Palette where it can be inserted into your process flows. It can be added as an inline, single use instance within the process flow, or created as an external standalone component (`.xJavaScript` process file), which can be shared amongst multiple process flows.

For more information on developing scripts with the JavaScriptProcess component, see *OpenText Embedded Output Transformation Engine - Developer's Guide (VDTOTS-H-PTE)*.

JavaScriptProcess Properties

The following parameter is available for JavaScriptProcess:

ScriptSource	Designates the script source to execute. Double-click either the component in the Development pane or the parameter's Value in the Properties pane to display the JavaScript source editor where you can write the code to execute inside the process flow.

5.4.11 JmsProcess

The **JmsProcess** is the opposite of the JMS Event. It sends information to a Java Messaging Service (JMS) queue to be processed by another Java application. For more information, see "JmsEvent" on page 176.

Properties set pointers to Java Naming and Directory Interface (JNDI) Context and ConnectionFactory.

JmsProcess Properties

Parameters for the JmsProcess are as follows:

Name	Specifies the name of the JMS handler.
IsBinary	Specifies whether messages are sent as ASCII or as binary data.
JndiContext	Using the Java Naming and Directory Interface (JNDI) allows Output Transformation Server to use the built-in capabilities of Java runtime environments for Java messaging. You can set multiple JNDI Property name-value pairs.
Property	Each property can contain one name-value pair. Right-click to add or remove properties.
Name	Specifies the name of a property to be put in the outgoing JMS message.
Value	Specifies the value of the property named above.
ConnectionFactory	The connection factory is an administered object that a client uses to create a connection to the JMS provider. A destination is an administered object that encapsulates the identity of a message destination, which is where messages are delivered and consumed. Set these parameters for the Connection Factory and destinations:
LookupName	Specifies the Connection Factory name in the JNDI tree to connect to.
User	Specifies the login user name.
Password	Specifies the login password.

IsQueueConnection Factory	Specifies the type of connection factory: Queue or Topic.
Retries	Indicates the number of retries before the connection fails. The default value of 0 means an unlimited number of retries.
Destination	
LookupName	Specifies the destination name in the JNDI tree.
JMSProperties	Lists the configured JMS properties. Right-click to add or remove properties.
Property	Specifies the name of the JMS property.
JobVar	Specifies the job variable associated with this JMS property.
BufferSize	Specifies the size of the reused buffer to pipe the data to the output file.
ReliableTransfer	This process includes ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.12 JobVariableLoggerProcess

JobVariableLoggerProcess records values of job variables used in a project to the OpenText Output Transformation Server log, which is particularly useful for debugging purposes. You can opt to either write out log messages for specifically named job variables or all variables while also selecting the logging level.

JobVariableLoggerProcess Properties

The JobVariableLoggerProcess component has the following parameters:

LogLevel	Designates the log level of messages to write to the log. You can select from the following levels: • **DEBUG** • **INFO** • **CONFIG** (This log level is obsolete, but equivalents to the INFO level.) • **WARNING** • **ERROR** • **FATAL** • **SUMMARY** (This log level is obsolete, but equivalents to the INFO level.) • **EXCEPTION** (This log level is obsolete, but equivalents to the ERROR level.)  **Note:** When writing out to a log level, the log messages are also written for all other log levels below it. For example, opting to write to the ERROR log level will also create messages in the WARNING, INFO, and DEBUG logs.
LoggingVariables	Specifies whether messages for a particular variable type or only exclusively defined variables are written to the log. The following variable types are available: • **ALL**. Every system, engine, and job ticket variable. • **ES**. Only variables related to Output Transformation Server's engine and job tickets. • **DEFINED**. Only messages for the identified variables in the Variables parameter are logged.
Variables	Indicates the specific job variables to write to the log.  **Note:** To use this parameter, the LoggingVariables parameter must be set to DEFINED otherwise the Variables parameter is ignored.

5.4.13 JobVariableSetterProcess

JobVariableSetterProcess automates the process of setting multiple job variables from a variety of sources for your projects. Job variables can be obtained from constants, XDocs, or SQL query results.

5.4.13.1 JobVariableSetterProcess Properties

The JobVariableSetterProcess component has the following parameters:

ConstantJobVariables	Specifies the list of job variables to create from constants.
Name	Indicates the name of the job variable.
Value	Indicates the value of the job variable.
XDocJobVariables	Specifies the list of job variables to create from XDocs.
Name	Indicates the name of the job variable.
XDocName	Denotes the name of the XDoc that contains the value for the job variable.
MappingType	Designates whether the XDoc is held in an input or result map.
SqlJobVariables	Specifies the list of job variables to create from SQL query results.
Name	Indicates the name of the job variable.
PersistentStorage	Denotes the name of persistent storage where the query is run.
SqlQuery	Denotes the SQL query used to retrieve the job variable value.
Delimiter	Indicates the delimiter for the recordSet fields returned by the SQL query. The default value is a comma.
Header	Specifies whether to include a header with column names. By default, this is false.

5.4.14 JPSPrintProcess

The **JPSPrintProcess** prints through Internet Protocol Printing (IPP). Printers used with this method must be IPP compatible to ensure that the Java Print Service (JPS) will properly and directly output print jobs at the specified printer.

Note: Although IPP compatibility will ensure that your printer will function with the JPS, the print results are largely dependent upon the printer itself and will vary from printer to printer.

For detailed log info generated by JPSPrintProcess or any of its properties, refer to `all-debug.textreport` located in the OpenText Output Transformation Server log folder.

5.4.14.1 JPSPrintProcess Properties

The following parameters are available for JPSPrintProcess:

PrintService	Identifies the name of available print services installed on the machine. If the value is left blank or results in an error, the default printer will be used.
UsePrintFile	Indicates which input print data will be used. If true, the print file from the file system is used. Otherwise, print data from the input map will be used.
PrintFileName	Indicates the name of the print file. This can be set statically or dynamically through job variables. If specified statically, it is the absolute path of the print file. If specified dynamically, it is the value of the job variable.
MimeType	Indicates the type of document that is to be printed. Additionally, your printer must support this document type.
PrintAttributes	Allows you to control features related to printing.
JobName	Allows you to set a name for the print job.

Destination	Specifies the full file path used to indicate an alternate Uniform Resource Identifier (URI) destination for the spooled printer data. If the selected print service supports this attribute, the print data will be printed to a file. If the selected print service does not support this attribute and it is not compromised, the print data will be printed to a file as well. If, however, the selected print service does not support this attribute and it is compromised, the print data will still be sent to the printer device, but this attribute is ignored.
JobPriority	Indicates the priority level for scheduling the print job.
Copies	Specifies the number of copies for the print job.
PrintQuality	Specifies the print quality of the printed output.
Media	Indicates the medium that the job will be printed on.
Sides	Indicates the number of sides available for printing.
CompromisePrintRequestAttributes	Specifies how unsupported print request attributes are handled. If true, any print request attributes that are not supported by the print service will be ignored for the print job.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.15 MailProcess

The **MailProcess** will send an email message, with or without an attachment. The message may, for example:

- Inform you that a file has been processed.
- Send the file itself as an attachment.

Parameters set properties for connection and attachment.

MailProcess Properties

Parameters for the MailProcess are as follows:

From	Specifies the return address to put on the email message.
To	Specifies the email address(es) to send the mail to.
Cc	Specifies the email address(es) of anyone who should receive a copy of the email message.
Bcc	Specifies the email address(es) of anyone who should receive a blind copy of the email message.
Subject	Specifies the subject line for the email message.
Attachment	Specifies the full filename of any attachments.
Port	Specifies the number of the port for accessing the email server.
Host	Specifies the name of the email server.
Timeout	Specifies the number of milliseconds before timeout.
IsAuthRequired	Check whether recipient SMTP server requires authentication.
User	Specifies the email server user login.
Password	Specifies the email server user password.
Smtp_auth_mode	Specifies standard or secure mode for mail server authentication.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.16 IBM WebSphere MQProcess

The **IBM WebSphere MQProcess** (MQProcess) sends an XDoc to a designated destination queue or topic. A *queue connection* needs the queue to be configured, while a t*opic connection* requires the topic to be set up in advance. This component places messages on a specified queue, such as the input queue.

To use the MQProcess, you need to implement a process flow that contains the process.

MQProcess Properties

Parameters for the MQProcess are as follows:

Name	Specifies the name of the WebSphere MQ handler.
IsBinary	Checks whether or not messages should be sent in binary mode, rather than text mode.
ServerConnection	Gives details for the server connection. The MQProcess uses the details from the local queue if this parameter is left empty.
Host	Identifies the remote host for the server connection. For local connections, leave this parameter empty.
Port	Identifies the remote port for the server connection. The default port is 1414. For local connections, leave this parameter empty, although it may have a value that will be ignored.
Channel	Identifies the server connection channel for the server connection. For local connections, leave this parameter empty, although it may have a value that will be ignored.
MQManager	Specifies the queue manager on the server (local or remote).
Name	Specifies the name of the queue manager on the server (local or remote).
User	Specifies the queue user (optional).
Password	Lists the queue user's password (optional).
IsQueueConnection	Specifies whether this is a queue (True) or topic (false) connection.
Retries	Provides the number of retries if queue connection fails.
Destination	Lists the name of the queue (local or remote). Contains a set of WebSphere MQ destination objects, such as topics and queues.
Name	Identifies the Destination name in WebSphere MQ.
MqProperties	Maps the value of a job variable to a WebSphere MQ property.
Property	States the name of the WebSphere MQ property.
JobVar	Identifies the name of the job variable that will store the WebSphere MQ property value.
BufferSize	Specifies the size of the reused buffer to pipe the data to the output file.

ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

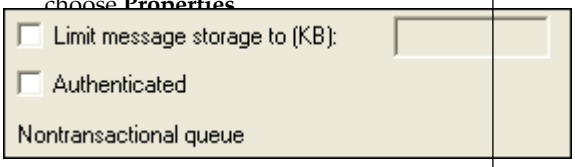
5.4.17 MSMQProcess

MSMQ is a Microsoft message queue utility. The **MSMQProcess** uploads files to the MSMQ.

For information about initializing and preparing MSMQ to be used with Output Transformation Server, see "Overview of MSMQ Connectors" on page 156.

5.4.17.1 Configuring MSMQProcess

Name	The name entered must be the same as the MSMQ message queue to which this process will load files.
DllQueuePath	Enter the full Windows path to the OpenText-supplied `MsmqJava.dll` file. If this path is left blank, Output Transformation Designer will look for the DLL in the Java library path (`C:\Java\libs\`). The installer places the file in the `<install_home>\lib\dll` folder.
IsBinary	Select if messages have to be uploaded in binary mode. Clear if messages are to be uploaded as text.
MqHost	Enter the computer name or IP address of the machine where the MSMQ message queue is configured. If this is the same machine as the Output Transformation Server application, enter localhost.
Destination	Enter the details of the configured target queue to which messages will be uploaded. The details entered must match the configurations in Windows exactly.

QueueTopic	Enter the type of queue. This will be private or public (see the MSMQ documentation), unless you have defined a custom queue type. **Note:** If you have created a customized URL for the queue connection, enter the URL here and do not enter anything at Name (below) or MQHost (above). If a custom URL is entered here, it will override anything entered in Name or MQHost. The URL would be in the format: `url= Direct= OS:machinename\\private$\ \queue` or `url= Direct=TCP:121.0.0.1\ \private$\\queue`
Name	Enter the exact name of the queue.
Transaction	Specify the transactional type of the queue. Choose **None** for a non-transactional queue. Choose **Single** for sending messages to a transactional queue. If you do not know if the queue is transactional: 1. Locate the queue in **Control Panel** > **Computer Management** > **Services and Applications**. 2. Double-click the queue, or right-click and choose **Properties**.  If the queue was configured as transactional and to work within Microsoft Transaction Server (MTS), choose **MTS**. If the queue was configured as transactional and to work with XA-compliant transactions, choose **XA**.
Reliable Transfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.17.2 MSMQ Job Variables

Once the message is uploaded to the MSMQ by this process, the following variables will be populated and available for use in the process flow that follows this component:

Variable name	Value
Label	The MSMQ message label.
CorrelationID	The MSMQ message correlation ID.
XdocToSend	The content of the MSMQ message.

5.4.18 PayloadSplitterProcess

PayloadSplitterProcess is a runnable component that can be included in a process flow to iterate through each ebXML payload received by EbXmlEvent for creating input data for the next process in the process flow.

For more information, see "EbXmlEvent" on page 166 and *OpenText Output Transformation Server ebXML Messaging - User Guide (VDTOTS-H-UXM)*.

5.4.19 PdfDsrProcess

The **PdfDsrProcess** (Data Storage Reduction) is an OpenText proprietary PDF compression facility that significantly reduces the long-term storage requirements for large collections of PDFs files. The process identifies and removes all common resources from a collection of PDF files, and stores them in a binary resource file. This means that in a collection of 10,000 PDF files, only one copy of a recurring resource such as a font, overlay or image is stored in the resource file, instead of multiple times in many, or perhaps all, of the documents. The remaining parts of the PDF after the common resources have been removed can be compressed using flate compression to further reduce the file size.

This Output Transformation Server component works with existing PDF files created by OpenText processes or by other programs. If files have been encrypted, the component can be configured to decrypt the files automatically.

Once the PDF DSR component runs, there will be 4 individual files in the output directory, which can then be loaded into a repository such as IBM OnDemand:

- the individual PDF files will be saved as one stacked PDF DSR file.
- the resources will be saved in a binary file.
- an index file.
- a statistics file.

Caution

Do not alter the output files. The index, DSR, and resource file contents are interdependent and modifying any file may compromise the integrity of the other files.

Before running the PdfDsrProcess on a large input file, configure and test it with a selective sample to determine the best configurations for your input. The parameters and statistics file explanations below will help you make the best adjustments for the optimal file size reduction.

5.4.19.1 PDF DSR Input XDocs

The process requires one incoming XDoc called InputIndex. This is generally created by a previous step in the process flow and is an IBM OnDemand index file pointing to all individual PDFs (stacked or separate) that will be processed.

5.4.19.2 PdfDsrProcess Properties

- **InputPath**. Indicates the full path of the directory containing the PDF files. Although the PdfDsrProcess component only supports OnDemand indexes as input, it can produce either OnDemand or ES Repository-type index files as output, which is indicated by the IndexType parameter. Note that this parameter is ignored if the CMOD index field `GROUP_FILENAME` already contains a valid file path location since that index field is referenced first.

  Also, if you are passing multiple PDFs through an input map PdfFilesToDsr, both this and the FileFilter parameter are ignored.

- **FileFilter**. Specifies the filter to use to gather the files from the InputPath parameter that are included in the batch. The FileFilter parameter is used only when no input mappings are supplied. When using this parameter, typically your input PDFs do not need to be stacked, however, if you have an input index that is passed through from an input map then the input PDFs must be stacked. Outputted index files from this parameter do not contain any properties.

  If you are passing multiple PDFs through an input map PdfFilesToDsr, both this and the InputPath parameter are ignored.

- **OutputPath**. Enter the full path to the directory where the four results files will be written.

- **OutputIndexName**. Enter a name for the result index file. This file will be created in the OutputPath directory.

- **IndexType**. Designates whether an **IBM OnDemand** or **Xenos Repository** index type file is generated.

- **OutputStackedName**. Enter the name of the resulting stacked output file. This file will be created in the OutputPath directory.

- **OutputResGroupName**. Specifies the file path directory or resource name to use for your resource file outputs, separate from the output directory used by index files in the OutputPath parm. Typically, both output types can be stored in the

same directory but there may be special circumstances where the resource files need to be put in an independent folder, such as to avoid storage duplication or if you need only the resource files for back up purposes. If you use an absolute file path value, then the resource files are stored here instead of the directory indicated in the OutputPath parameter (the folder is created automatically if it doesn't exist). If only a resource name is provided, then the resource file is generated with this name in the OutputPath's directory. If the value is left empty, the output is directed to an XDoc in the result map instead.

- **StatsName**. Enter the name of the stats file. This file will be created in the OutputPath directory.
- **OutputBufferSize**. The default buffer size of 5 kilobytes is sufficient in most cases. If you find the process is running much more slowly than expected, or is using too much of your system resources, modify the buffer size incrementally to see if the issue is resolved.
- **CreateOutputDir**. Enable if the OutputPath defined above does not currently exist.
- **Args**.
  - **Threshold**. Enter a value of 1% to 100%. This is the frequency with which a resource must occur within input documents for the resource to be removed. For example, if the threshold is set to 100%, a resource must exist in every input document to be removed and placed into the resource file. Unless there is a extraordinary overlap of resources used in every input PDF, a threshold of 100 may not result in a significant file size reduction since there will be too few common resources. Therefore, the time and effort involved in the DSR process would not be justified. However, setting the threshold to a low value, such as 20% may make the process slow and use excessive system resources for minimal gain. The optimal value will usually be between 50 and 80 per cent, depending on the input.
    The statistics file will be invaluable in tweaking this value to get the best cost/benefit ratio in terms of file size and processing time.
  - **DoCompression**. Enable if you want to use flate compression on the remaining readable text in the PDF DSR files, which can potentially further reduce the storage size by 10 to 20 percent.
  - **DoDecryption**. Enable if the input files are encrypted. Files will not be encrypted again after processing. (If a PDF file was encrypted with a read password, the PDF DSR component cannot decrypt the file and it will be skipped.)
  - **StreamSignatureMinSize**. The resource markers (the reference markers that are inserted into the file in place of a resource) are 20 bytes, so the minimum size of the stream data resource to consider for extraction must be > 20 bytes. The benefit of extracting the resource is not significant until > 100 bytes.

```
Total docs processed: 46
Total docs reduced: ALL
Original size: 12514477
Descrypt size: 12343026 99%
DSR size: 3087114 25%
Compressed size: 1300039 10%
Resource count: 7
Resource  size: 251060
Parse   time: 2078
Decrypt time: 1388
Analyze time: 0
Write   time: 203

ResID, Length, UsageCount, DocCount, Precentage
1,      150,    46,     46,     100%
2,      22783,  46,     46,     100%
3,      52992,  46,     46,     100%
4,      85829,  46,     46,     100%
5,      1192,   46,     46,     100%
6,      150,    46,     20,     43%
```

5.4.19.3 l

l help to
for the data in

)SR Process
field, to

Total docs processed is the number of documents in the input.

Total docs reduced is the number of documents that were reduced using DSR. If this number is significantly less than the total docs processed:

- check that the threshold was not set too high. The resource details section can help determine an appropriate threshold.
- check that the documents are not read protected.
- ensure that the input documents are documents with some similarity that are grouped together logically.

The **original size** and **decrypt size** are the total bytes of the input documents. The percentage displayed at decrypt size is the percentage of the original size. If decryption is not enabled, decrypt size will not appear in the stats file.

The **DSR size** is the size of the stacked output PDF DSR file in bytes. The percentage is the percentage of the original size. Ideally, the file would be no larger than 30% of the original size. If it is larger, for instance, 50% of the original size, consider adjusting the threshold to remove more resources.

The **Compressed size** is the final size after flate compression, if enabled.

Resource count is the number of resources removed and added to the resource file. If this number is low, consider adjusting the threshold to remove more resources. However, if this number is low and the DSR size is less than 30% of the original, there may simply be few but large resources used across the input files and not adjustments are required.

Resource size is the size, in bytes, of the created resource file.

Parse time, **decrypt time**, **analyze time**, and **write time** are the times in milliseconds that the total conversion took to complete. The sum of these four times is the total processing time.

The resource section details the resources found in the input PDF documents:

ResID begins at 1 with the first resource extracted and increments for each resource.

Length is the byte size of the resource. Keep in mind that each resource marker is 20 bytes, so if many small resources are being extracted, the benefit may not outweigh the cost. The Stream Signature Min Size may need to be adjusted.

Usage count is the number of times the resource was used throughout the input. This number may exceed the number of documents in the input if a resource was used more than once in a document.

The **Doc count** is the number of documents in which the resource was located and extracted. If the threshold is 100%, the doc count will always equal the total docs processed.

Percentage refers to the percentage of documents in which the particular resource was found. This column is essential in determining the appropriate threshold. In the example below, resource 7 occurs in only 43% of documents, however it is a relatively large size and its extraction could significantly reduce the overall size of the output. Therefore, setting the threshold at 43% would be beneficial. However, resources 8 and 9 are very small and little benefit would be gained by extracting these resource from 30% of the input. If the sizes of the resources was reversed (small resource 7 in 43% of documents but large resources 8 and 9 in 30% of documents each), it may be worthwhile to move the threshold to 30%.

```
ResID, Length, UsageCount, DocCount, Precentage
1,      150,    46, 46, 100%
2,      22783,  46, 46, 100%
3,      52992,  46, 46, 100%
4,      85829,  46, 46, 100%
5,      1192,   46, 46, 100%
6,      150,    46, 46, 100%
7,      67882,  20, 20, 43%
8,      206,    14, 14, 30%
9,      206,    14, 14, 30%
10,     1723,   10, 10, 22%
11,     1701,   10, 10, 22%
12,     1674,   10, 10, 22%
13,     1736,   10, 10, 22%
14,     1611,   10, 10, 22%
15,     1655,   10, 10, 22%
```

5.4.19.4 Uploading Result Files to a Repository

Once the process has completed, the files are ready for uploading to a supported repository for long-term storage.

By default, the index file is in IBM OnDemand format and can be readily uploaded to CMOD using the ARSLOAD process. Prior to uploading, the PDFDSR file type must be defined in CMOD.

Any other repository that is compatible with the IBM OnDemand index format can be used for storage.

Whichever repository you choose to upload the files to, the results are intended as long-term storage. Using the index file, individual documents can be extracted for

viewing, if required. The viewed PDF would be deleted afterwards, and the document would still exist in the PDF DSR file.

5.4.20 PdfMergeProcess

The **PdfMergeProcess** joins multiple PDFs residing in a MultiXDoc or a directory, or multiple XDocs into a single merged PDF file, along with a result XDoc containing a semicolon separated list of the files that were merged.

The input for the PdfMergeProcess can be:

- A directory of PDF files, or
- A MultiXDoc

The result will be two XDocs:

- **MergedPDF** is the result XDoc containing the merged PDF documents
- **ListOfMergedPDF** is the result XDoc containing the semicolon separated list of PDF filenames or input MultiXDoc keys that were merged

PdfMergeProcess Properties

The PdfMergeProcess component has the following parameters:

- **SourceType**. Choose **DIRECTORY** if the input is a directory containing multiple PDF files, or **MULTIXDOC** if the input is a MultiXDoc.
- **Source**. Click the **ellipsis** and navigate to the location of the source directory or file.
- **MergeLimit**. Enter the maximum number of files that can be contained in one merged PDF output file. Once this limit is reached, another file is created and the resulting MergedPDF and ListOfMergedPDF XDocs become MultiXDocs.
- **DeleteMergedFiles**. If selected, the source files are deleted upon successful completion of the merge process.

5.4.21 ProcessFlow

The **ProcessFlow** component links various processes together and executes them in a specific order.

For instructions on how to configure the ProcessFlow component, see "Setting Up a New Process Flow" on page 68.

5.4.22 RuntimeExecProcess

The **RuntimeExecProcess** is a process that runs a command line application. Some examples of applications that this process can execute are `.exe`, `.bat`, `.com`, and `.sh` types.

RuntimeExecProcess Properties

The following parameters are set for the RuntimeExecProcess:

- **Command**. Contains the name of the executable file, including absolute path and extension.
- **Arguments**. Lists the arguments to pass to the executable.
- **WaitForProcess**. Specifies whether or not to wait for the process to complete before returning to the calling process.
- **StdOutLog**. Determines where to write all standard console output. When choosing to persist this information it can be written to either a log file or to an XDoc.
- **StdErrLog**. Determines where to write all standard error console output. When choosing to persist this information, it can be written to either a log file or to an XDoc.
- **WriteToStdIn**. Defines whether or not to use the input XDoc as standard input. When this option is selected, the incoming source XDoc will be used as standard input for the executable.
- **XDocVarMap**. Contains a list of mappings that specify how to send a source file (as an XDoc) to the executable and how to retrieve a target file produced by the executable (as an XDoc).

The screen shot below shows the configuration for running a command line application called compressIt.exe, which requires a source file and compresses it to a destination file. The source file is referred to as the job variable ${in} and the destination file is referred to as ${out}. The <*XDocVarMap*> parameter is then used to map those job variables to an XDoc in either the input or result map.

5.4.23 SocketProcess

The **SocketProcess** sends data to a socket. Parameters define connection properties.

SocketProcess Properties

Parameters for the SocketProcess are as follows:

Response	Checks whether this process requires a response from the socket server it connects to before time out.
BufferSize	Specifies the size of the reused buffer to pipe the data to the output file.
Port	Specifies the port from which to send the socket stream.
Host	Specifies the socket server host name.
Timeout	Specifies the number of milliseconds to wait before timing out (per scan).
IsBinary	Select this option here if data is to be treated as binary, rather than text.
Header	Specifies the message length in the message header.
HandlerOutputBuffer	Specifies the input buffer size for socket handler.
Terminator	Specifies message terminator by ASCII HEX code, such as 0x1c;0x0d;.
ReliableTransfer	This process requires ReliableTransfer functionality. For more information, see "ReliableTransfer" on page 278.

5.4.24 TiffAppender

The TiffAppender component is used in a Output Transformation Server process flow to concatenate two input TIFF files into a one output TIFF file.

The inputs are two input TIFF files, supplied by a Get File Process or any other process that can produce a TIFF file.

Once the TiffAppender Process combines the files into one, the resulting TIFF file can be used by another component. For example, you can output the file to the file system using a File Process component, or to an FTP site using the FtpProcess component. Any component that can use a TIFF file input can be used after the TiffAppender Process.

Related Links

- "GetFile" on page 198

- "FileProcess" on page 198
- "FtpProcess" on page 217

5.4.24.1 Using the TiffAppender Process

In the following example, two TIFF files will be used as input using the GetFile Process component, they will be combined using the TiffAppender Process component, and then the resulting TIFF file will be output to the file system using the File Process component.

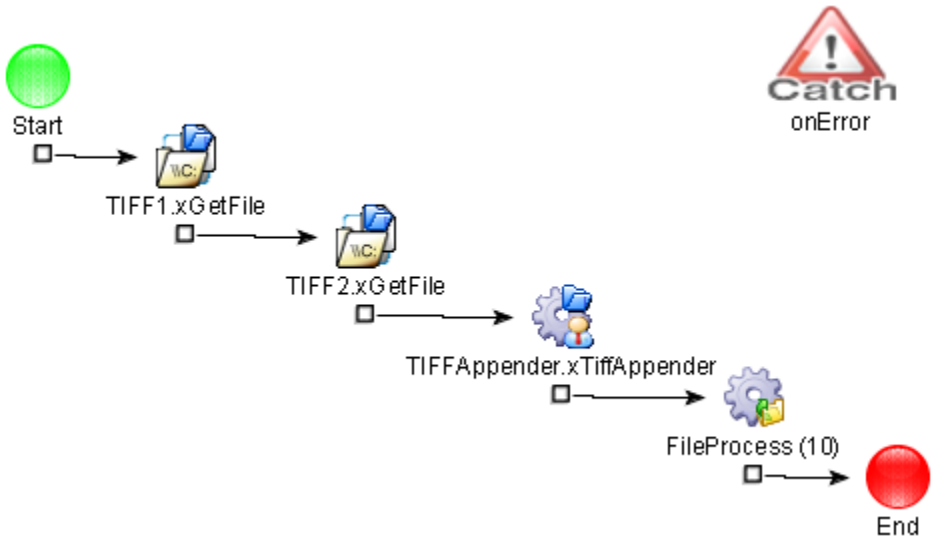


Figure 5-13: Example TiffAppender process flow

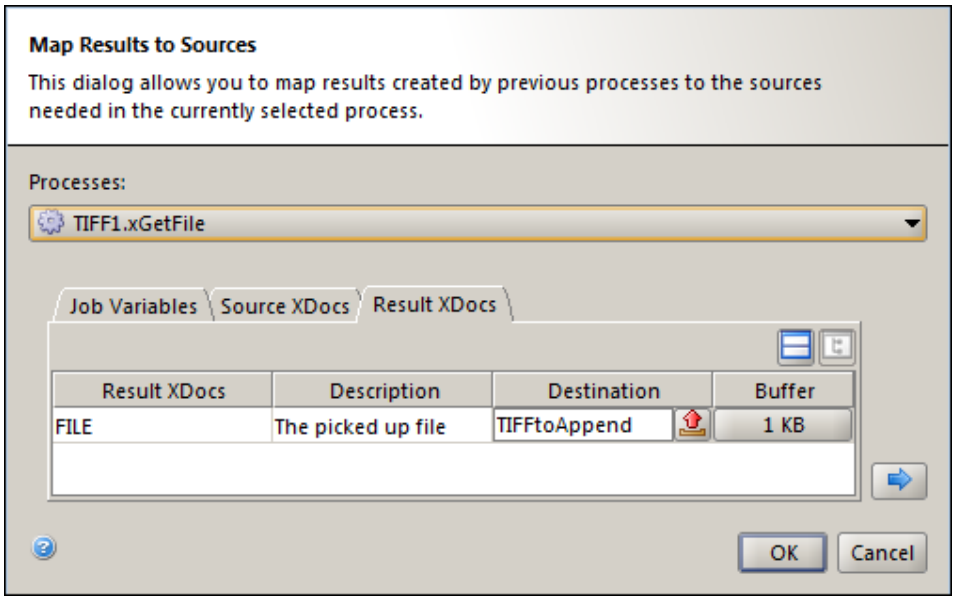
Related Links

- "GetFile" on page 198
- "FileProcess" on page 198

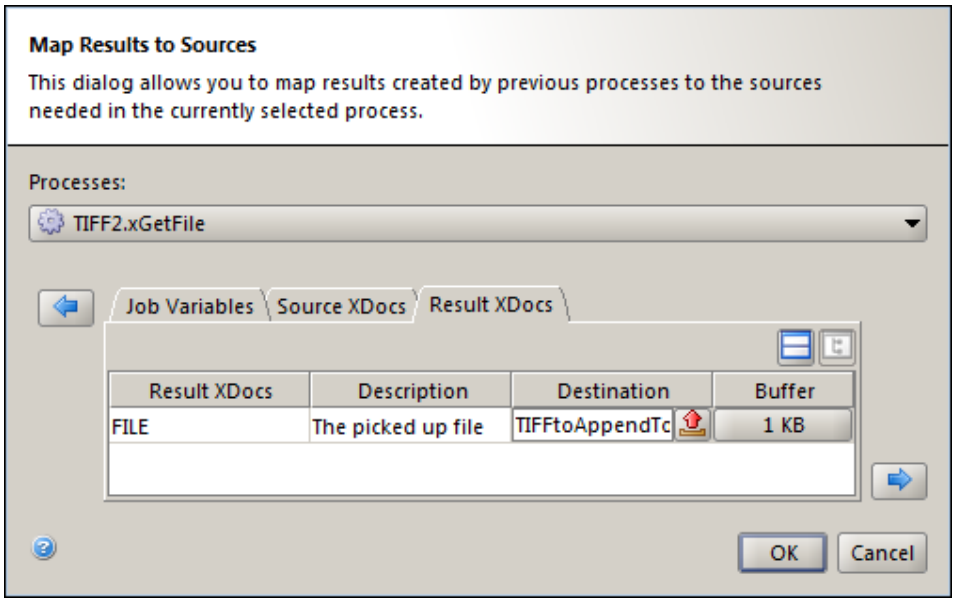
Mappings

1. Right-click a component and choose **Show Mappings**.
2. In the **Source and Result Mappings** dialog, enter a Result XDoc **Destination** name for each of the GetFile processes.

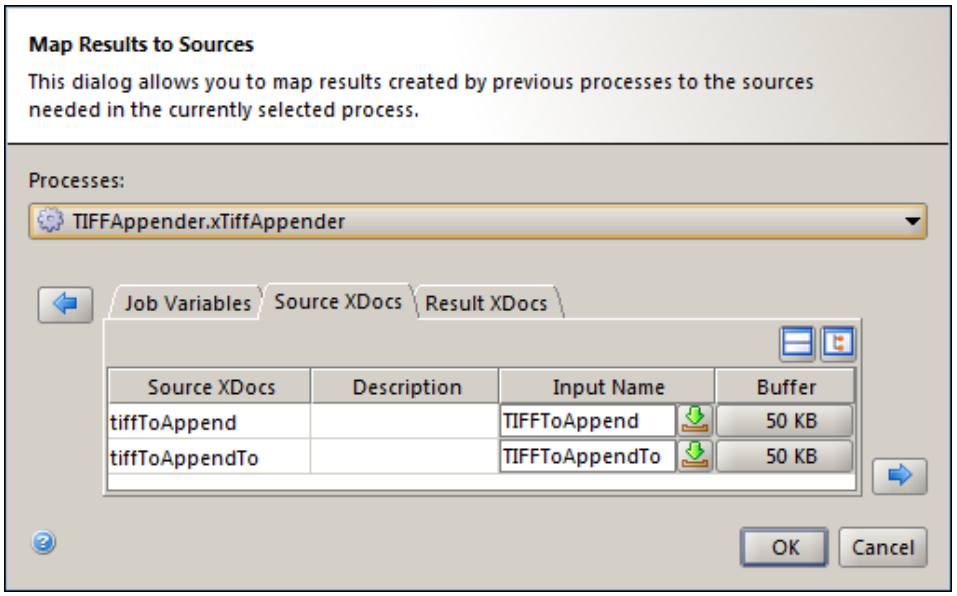
   The first example process uses TIFFtoAppend as the Destination.



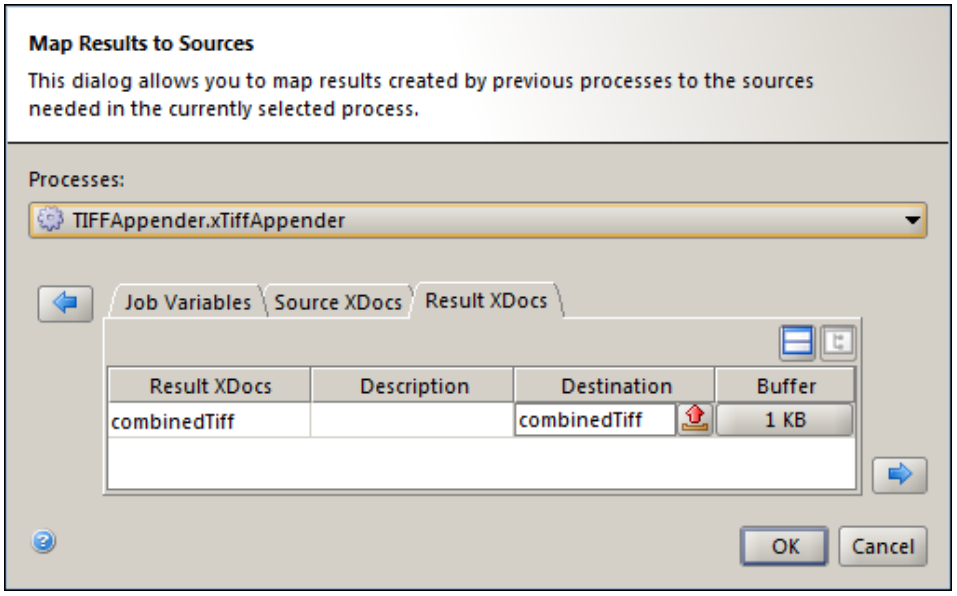
The second example process uses TIFFtoAppendTo as the Destination.



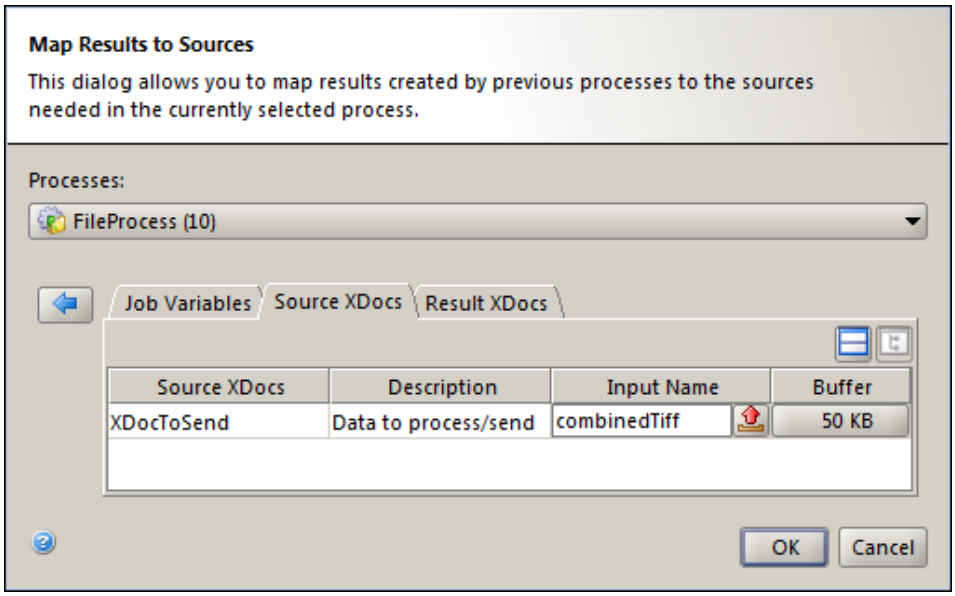
3. Enter both of the names from Step 2 as the Source Xdocs **Input Names** for the TiffAppender component.



4. Enter a Result XDoc **Destination** name for the TiffAppender component.



5. Enter the name from Step 4 as the Source Xdocs **Input Name** for the File Process used to send the combined TIFF result to a file.



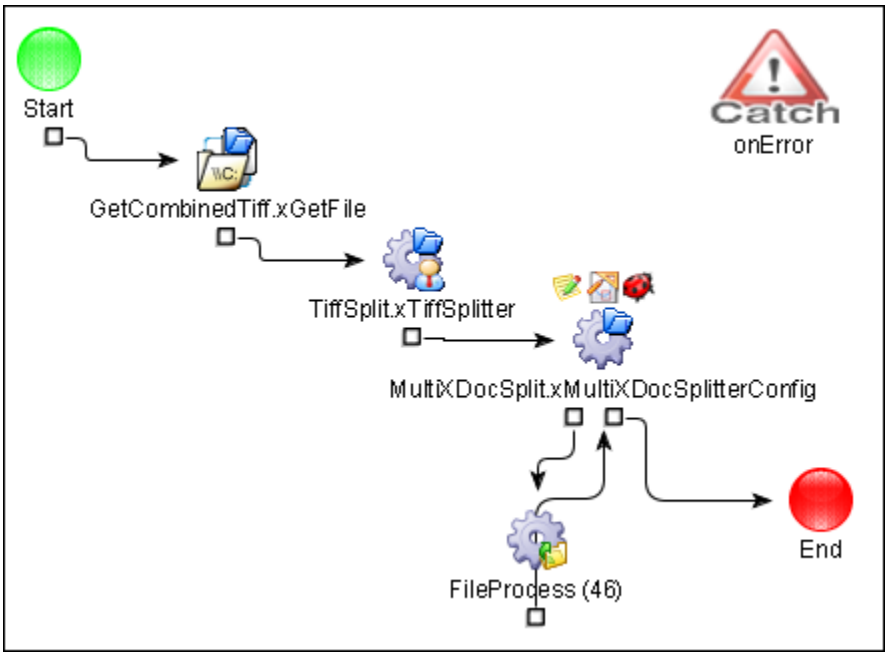
5.4.25 TiffSplitter

The TiffSplitter process separates a multiple-page TIFF file into multiple single-page TIFF files. The output of the TiffSplitter process is a multiXDoc.

If the input file is a single-page TIFF, the resulting multiXDoc will be one page.

Using the TiffSplitter Process

The following example is a Output Transformation Server process flow with a TiffSplitter component following a Get File input process and preceding a MultiXDoc Splitter process. It will use a FileProcess to write the output to the file system.



The input Get File process supplies the TIFF file.

The TiffSplitter process has no configurable parameters.

The output is a multiXDoc, which is processed by a MultiXDocSplitter component.

In this example the MultiXDocSplitter component is followed by a FileProcess to write the output to the file system. Because the MultiXDocSplitter will write out more than one TIFF result file, the Write Mode of the FileProcess must be set to APPEND_UNIQUEID.

Related Links

- "MultiXDocSplitter" on page 206
- "GetFile" on page 198
- "FileProcess" on page 198

5.5 Sample Test Components

Along with the OpenText Output Transformation Server core components, such as File Event, Mail Event, File Process, and Mail Process, there are also some sample components you can modify. The **ComponentDef** files and all the necessary source code are included in the `<install_home>\intitialFiles\common\com\xenos\framework\bpengine\test` directory. For more information, see "Component Definitions" on page 83.

All sample components begin with the word *Test*. You can modify the source files and recompile. The classes are generated in the directory:

```
 <install_home>\initialFiles\common\_classes\com\xenos\framework\
bpengine\test
```

In order for the new class files to be available on a server running Output Transformation Server, the new class files need to be redeployed. To do this:

1. Locate the new class file in the classes directory.
2. Right-click the file and select **Deploy**.

Caution

Changing the code of any Test*Process behavior will impact the sample process flows using it. An example is the Isv2File.xProcessFlow's start process, which calls TestSimpleProcess.

5.5.1 ContentIdentifier

The **contentIdentifier** component allows you to route your transaction flow based on the content of a message, with assistance from the JobVariableRouter . The file types are detected by the component by reading a portion of the XDoc. File types that contain routing options include XML, CSV, EDIX12, EDIFACT and SWIFT. Once the content is identified, the variable set in the **JobVarToPopulate** parameter is used to route the data within the Job Variable Router. For more information on the JobVariableRouter, see "JobVariableRouter" on page 201.

Additionally, EDIX12 and EDIFACT permit routing based on the transaction type, so the variable will be set as X12-*xxx* or EDIFACT-*xxx*, where *xxx* represents the transaction type.

5.5.1.1 Configuring the contentIdentifier

1. Choose the data type(s) to enable.

   The available options are XML, CSV, EDIX12, EDIFACT and SWIFT. When a selected data type is detected, the job variable is set to that data type and the data is routed to that data type flow.

   Example: The **HandleXML** option is selected. When XML data is detected, the job variable is set to **XML** and the data is routed to the XML flow by the JobVariableRouter.

2. For **JobVarToPopulate**, enter the job variable to populate with the message type.

   The default value is **docType**, but you can edit this by clicking the **Ellipsis** button and entering a different value.

   Example: HandleXML is selected and the job variable **routeToTake** is entered. When XML data is received, **routeToTake** will be set to **XML**.

3. Once you have selected the data types you want to use, add the JobVariableRouter as the next process in the flow. For more information on the JobVariableRouter, see "JobVariableRouter" on page 201.

Note: A sample working flow can be found in `_sample\OutputTransformation\simpleContentRouter\testSimpleRouter.xProcessFlow` within the main file system. You can also view the Java source of the simpleContentRouter under `\com\xenos\framework\bpengine\router\ContentIdentifier.java` in the main file system.

5.5.2 TestDynamicProcess

The **TestDynamicProcess** compares the value of a variable, named in the JobVar parameter, with a set of values set in the CompareList > TestValue parameters. When the values match, the output is routed based on the named condition.

TestDynamicProcess Properties

The parameters to TestDynamicProcess are as follows:

JobVar	Specifies the name of the variable whose value will be compared.
CompareList	Specifies the name/value pairs for the variable named in the <*JobVar*> parameter. You can set multiple CompareList TestValue/ConditionName pairs as needed for the number of possible values of the variable named in the <*JobVar*> parameter.
TestValue	Specifies the value to compare to.

ConditionName	Specifies a reference name for the condition where the value named above matches the value of the JobVar variable. This can be the name of the process to be invoked, but does not need to be. It is just a reference name.

You can dynamically add new outputs to this component. Every time a new output is added, a new entry is added to the <*CompareList*>. This list consists of a collection of conditions. Each condition is based on a <*TestValue*> and a <*ConditionName*>.

When `TestDynamicProcess` is invoked, the value of <*JobVar*> will be compared against all the <*TestValues*> in the <*CompareList*>; when a match is found, it will route the flow to the corresponding output.

5.5.3 TestEbXmlProcess

The **TestEbXmlProcess** is used to log ebXML metadata, which can be retrieved and subsequently stored in job variables for specific messages. It then writes the payload to a user defined directory.

For more information, see *OpenText Output Transformation Server ebXML Messaging - User Guide (VDTOTS-H-UXM)*.

5.5.4 TestIncrementProcess

The **TestIncrementProcess** increments a variable by one each time it runs.

TestIncrementProcess Properties

TestIncrementProcess has just one parameter:

JobVar	Increments the value specified in JobVar by one, when invoked.

5.5.5 TestSimplePausableProcess

The **TestSimplePausableProcess** prints a message and the value of a job variable into the log, then pauses.

TestSimplePausableProcess Properties

This process does the same as the `TestSimpleProcess`, except that when `TestSimplePausableProcess` is invoked it runs indefinitely, executing the `runComponent()` method over and over until it is either paused or cancelled. For more information, see "TestSimpleProcess" on page 248.

5.5.6 TestSimpleProcess

The **TestSimpleProcess** prints a message and the value of a job variable into the log.

TestSimpleProcess can be used for the following:

- Testing progress through a series of processes.
- Giving quick status reports.

TestSimpleProcess Properties

There are two parameters for TestSimpleProcess:

- JobVar
- WhatToSay

When invoked, `TestSimpleProcess` will write the value of <*WhatToSay*> to the log. It will also write the value of the job variable specified in <*JobVar*>.

The <*JobVar*> parameter uses variables localized to the job being run. The <*WhatToSay*> parameter also uses variables from the JobTicket, but it can use global variables as well; for example:

```
Hello World ${varName}
```

5.5.7 TestSplitProcess

The **TestSplitProcess** checks a value and routes the output to three separate processes, depending on whether the value is *greater than*, *equal to*, or *less than* a specified value.

In a payment system, TestSplitProcess might be used to determine if the transaction should be sent through normal processing, a refund system, or collections. For example, if the amount past due was greater than zero, it might route to a collection system, but if it was less than zero it might route to a refund system.

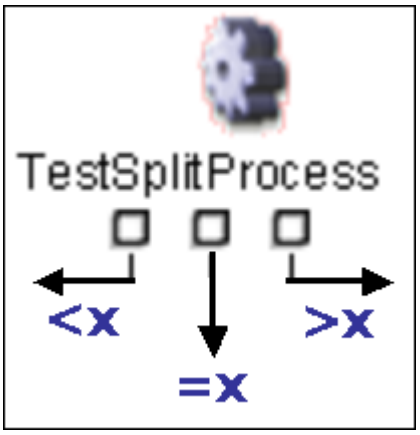


Figure 5-14: TestSplitProcess routing options

The three connections from the TestSplitProcess connect to processes that do the following:

- Execute when the job variable is less than the compare value.
- Execute when the job variable is greater than the compare value.
- Perform when they are equal.

However, the three connections do not have to be three separate processes. For example, the *equal* connector may point to one of the following:

- The same process as the *less than* or *greater than*.
- One process, and the *less than* and *greater than* point to another.

5.5.7.1 TestSplitProcess Properties

There are two parameters for the TestSplitProcess:

- *<JobVar>*
- *<CompareValue>*

When `TestSplitProcess` is invoked, it compares the value of *<JobVar>* to the value of *<CompareValue>* and, based on whether *<JobVar>* is *less than*, *equal to*, or *greater than* the *<CompareValue>*, it will route the flow to the corresponding output.

5.5.8 XDocJobVarConverter Process

The **xdocJobVarConverter** process converts from XDoc to Job Variable, and vice versa, when called upon.

5.5.8.1 xdocJobVarConverter Properties

xdocJobVarConverter has the following parameters:

SourceTarget	Specifies the conversion source and target. You can select from **JobVar2XDoc** or **XDoc2JobVar**.
NameToUse	Indicates the name of the job variable. This can denote either the job variable being converted to or from.
KeyToUse	Specifies the key for the XDoc to be either converted to or from.
MapToUse	Specifies the mapping to use.
InputMap	Indicates where the XDoc is to be retrieved as a conversion source.
ResultMap	Indicates where the XDoc is to be written to as a conversion target.

5.6 Services

Services are components which run in the background to provide overarching functionality to events and processes.

5.6.1 Alternate Text Manager

The Alternate Text Manager service allows you to connect to an Alternate Text Manager database in order to manage your alternate text resources. Alternate text is a vital component in making PDFs accessible since PDF documents often contain visual elements that screen reader technologies cannot interpret for users. Alternate text is provided for these elements so that screen readers can describe the visual content for visually impaired users. For example, instead of a logo's path and file name being read by the screen reader, you can define "company logo" as the alternate text to be read aloud.

The Alternate Text Manager service performs the following functions:

- Packages alternate text resources from Output Transformation Designer for uploading to the Alternate Text Manager database
- Administers the Alternate Text Manager database and its contents, allowing users to manage database connection information, tags, and alternate text resources

- Downloads resources from the database for use in Output Transformation Designer

For more information on alternative text and packaging alternate text resources using Output Transformation Engine, see *OpenText Embedded Output Transformation Engine - User Guide (VDTOTS-H-UTE)* and *OpenText Embedded Output Transformation Engine - User Guide (VDTOTS-H-UTE)*

5.6.1.1 Alternate Text Manager Component Properties

The following parameters are set for the Alternate Text Manager component:

- **ConnectionType**. Specifies the type of connection to use when accessing the database where Alternate Text resources are stored. You must also configure the associated parameters corresponding to the connection type. Select from either RepositoryConnectionParm, JdbcConnectionParm, or PersistentStorageConnectionParm.
  - **JdbcConnectionParm**. Uses JDBC (Java Database Connectivity) connection information.
    - **UserName**. Specifies the user name to use for connecting to the database.
    - **Password**. Specifies the password that corresponds to the database user name.
    - **JdbcDriver**. Indicates the JDBC driver class to use for the database.
    - **JdbcUrl**. Indicates the JDBC URL used to connect to the database.
    - **SqlDialect**. Specifies the SQL dialect for the database. The database administrator can provide the relevant information for your system.
  - **PersistentStorageConnectionParm**. Uses a persistent storage connection from the Persistent Storage Pool . When using this option, you must also specify certain hibernate mapping XML files in the system configuration file.
    - **PersistentStorageName**. Indicates the name of the persistent storage instance as defined in the system configuration file.
- **AuthenticationGroup**. Specifies the group to authenticate against. The default value is **AlternateTextAdmin**.
- **ShowSql**. Specifies whether or not to log SQL statements. By default, this option is disabled.
- **LockExpireTime**. Determines the amount of time, in minutes, that an image will remain locked in the Alternate Text Administration Console before the lock expires. The default value is 30 minutes.

Related Links

- "PersistentStoragePool Parameters" on page 29, especially "MappingResources" on page 33

- “System Configuration“ on page 17

5.6.1.2 Accessing Alternate Text Manager

The Alternate Text Manager is a web-based interface where Accessibility experts can create and configure written descriptions for graphical elements in a PDF/UA document for their visually-impaired clients. Alternate text is a requirement in Accessibility and PDF/UA specifications that permits assistive technologies to read out loud any visual aspects, while not affecting the appearance of documents for sighted clients.

There are a few separate parts involved in setting up Alternate Text Manager. You can manually create the deployment package, set up the database tables using the included database creation scripts, and configure the Alternate Text Manager component yourself, but a script that automates the process is available so that you can get started working with Alternate Text Manager as quickly as possible.

For more information on using the set up scripts, see *OpenText Alternate Text Manager - User Guide (VDTOTS-H-UAT)*.

5.6.1.3 Creating Alternate Text Manager Database Tables

For users who choose to manually configure the Alternate Text Manager component instead of using the Alternate Text Manager set up script, depending on your selected connection type then you may also need to create and populate the database tables that Alternate Text Manager requires to function.

If you are setting either **RepositoryConnectionParm** or **JdbcConnectionParm** as your connection type in the Alternate Text Manager component, you must run one of the SQL scripts included with your installation to create the tables. There are separate SQL scripts available for each type of supported database, which are stored in the `<install_home>\install\<version>\config\sql\atm` directory. To create the tables in the database, you must run the appropriate SQL script for your database type through your database client. The following scripts are available:

Table 5-1: Creating Alternate Text Manager Database Table Scripts

SQL Script File Name	Database Type
`atm-oracle.sql`	Oracle
`atm-postgres.sql`	PostgreSQL
`atm-sqlserver.sql`	Microsoft SQL Server

Note: Oracle users are required to submit a `COMMIT` command after running the script to end the current transaction and save all changes.

If you are using **PersistentStorageConnectionParm** as your connection type and have added the hibernate mapping resources, you do not need to create the database tables with these scripts since Output Transformation Server automatically creates

and populates the tables while initializing the persistent storage pool. For more information on the hibernate mapping resources to add, see "MappingResources" on page 33.

5.6.1.4 Updating Alternate Text Manager Database Tables

Users with existing Alternate Text Manager database tables from a pre-version 16.3 release must update their databases before they can be used with any subsequent versions. Some update SQL scripts are available to help you make the upgrade to your database without losing any of your stored data. There are separate SQL scripts available for each type of supported database, which are stored in the `<install_home>\install\<version>\config\sql\atm` directory. To update the tables in the database, you must run the appropriate SQL script for your database type through your database client. The following scripts are available:

Table 5-2: Updating Alternate Text Manager Database Tables

SQL Script File Name	Database Type
`atm-upgrade-oracle.sql`	Oracle
`atm-upgrade-postgres.sql`	PostgreSQL
`atm-upgrade-sqlserver.sql`	Microsoft SQL Server

5.6.2 Output Transformation Service

Placing a **Output Transformation Service** component into a process flow starts Output Transformation Engine. From there, you can run Output Transformation Engine projects within the larger Output Transformation Server process flow.

5.6.3 FileCleanup

The FileCleanup component is used to move or delete files and subdirectories in a specified directory after a given period of time. This service is useful when temporarily storing files for short periods of time, such as customer statements requested on demand, or files stored short-term post processing, which would not be deleted until the integrity of the output process was verified.

This component is a service. Once it starts, it will watch the specified directory until stopped, or until Output Transformation Server is shut down. In production, it can be started and stopped from the Services screen of the Server Tools. The service can also be configured to auto-start by configuring it in the System Configuration file, or by selecting the option on the Services screen of the Server Tools.

To make changes to the component configurations:

1. Stop the service.
2. Make the desired changes.
3. Restart the service, saving your changes when prompted.

Related Links

- "Services" on page 284
- "System Configuration" on page 17

5.6.3.1 FileCleanup Properties

The following parameters are set for the FileCleanup component:

- **DaysToLive**. Enter the number of days to keep files in the directory. If you want to retain the files for less than a day, enter 0 and use **HoursToLive** or **MinsToLive** instead. This field can be used in conjunction with HoursToLive and MinsToLive.
- **HoursToLive**. Enter the number of hours to keep files in the directory. This value can be used in conjunction with HoursToLive and MinsToLive.
- **MinsToLive**. Enter the number of minutes to keep files in the directory. This value can be used in conjunction with HoursToLive and MinsToLive.
- **ScanFolder**. Specifies the directory to watch for files. Once a file is added to the directory, the days, hours, or minutes to live countdown begins.
- **Filter**. Specifies the files, file type(s), or subdirectories to remove from the directory. This field accepts regular expressions so you can specify numerous inclusions and exclusions, as required. Enter * (asterisk) to remove all files.
- **ProcessAction**. Choose **Move** or **Delete**.
- **DestinationPath**. If you chose **Move** for **ProcessAction**, enter the path to the new directory where the files will be stored.

5.6.4 HttpService

The HttpService listens to incoming HTTP and SOAP requests and hosts the HTTP Events that submit jobs to a process flow. For more information, see "HttpEvent" on page 175.

5.6.4.1 HttpService Properties

The following parameters are set for the HttpService:

- **Listener**.
  - **Port**. Specifies the HTTP port number to listen on.
  - **Type**. Set to **socket**, **AJP** (Apache JServ Protocol), or **JSSE** (Java Secure Socket Extension).
  - **Keystore**.
    - **File**. Specifies the name of the file where the authentication keys are stored.
    - **Password**. Specifies the password for accessing the keystore file.

    - **KeyPassword**. Specifies the password for each private key.
    - **AuthenticateClient**. Checks if client certificate authentication is required.

- **Context**. Context is used to specify the regular web applications deployed into the HTTP server. WebContext is used to specify Axis web service applications deployed into the HTTP server which has an embedded Axis server (webapps/axis) as web service container:
  - **ResourceBase**. Specifies the name of the web application directory or WAR file.
  - **Path**. The context path specification, which must be in the form `/` or `/path/*`.
  - **VirtualHost**. (Optional) Virtual Host name.
  - **Servlet**.
    - **RequestSource**. Checks if the requested source is secure (**socket**=HTTPS or **socket/html**=HTTP).
    - **Name**. The servlet name.
    - **ClassName**. The class name of the servlet.
    - **Path**. A path specification to map this servlet to.
- **WebContext**.
  - **ResourceBase**. Specifies the Web service application directory or WAR file.
  - **Path**. The context path specification, which must be in the form / or /path/*
  - **VirtualHost**. (Optional) Virtual Host name.
- **InvocationDescription**. The description to appear in the JobStats indicating how the process flow was started; defaults to the name of the event configuration file.
- **Comment**. (Optional) A comment to appear in JobStats when the process flow is submitted.

5.6.5 JobMonitor

The **JobMonitor** service can keep watch over a process flow or event to ensure processes are happening as scheduled. For example, if a process flow is supposed to commence every weekday at 11 AM and for any reason it does not start, the Job Monitor can automatically initialize a separate process flow to carry out a series of processes such as send an email alerting an administrator of the issue.

A JobMonitor alert can be configured to track either time periods, time frames, or both. Creating an alert that combines both options is the most dependable method of ensuring that external systems which communicate with Output Transformation Server are still running and communicating appropriately.

5.6.5.1 Creating a JobMonitor Service Component

You must have a JobMonitor service component created before you can perform any configuration.

To create a new JobMonitor service component:

1. Create a new **JobMonitor** component using the **New Component Wizard**. To locate the JobMonitor component type, navigate to **Services** > **JobMonitor**.

   After you have finished creating the new component, the **JobMonitor** utility opens on a new tab in the **Development** window.

   Note: For more detailed information on establishing new components, see "Creating a Component" on page 84.

2. You must set JobMonitor to automatically launch whenever you start up the server. From the **Servers** tab, open the **Services** window and locate the **JobMonitor** component in the list. Ensure that JobMonitor is set to **Auto Start**.

   The JobMonitor is activated and ready to be configured.

   Note: In order to see the Services option in the Servers tab, you need to have deployed a server. For more information on deploying a server, see "Server Deployment" on page 279.

5.6.5.2 Creating a JobMonitor Alert

Note: Prior to creating a JobMonitor alert, you must have the process flow to run in case of a triggered alert already configured.

To create a new JobMonitor alert:

1. From the **File Systems** window, locate your **JobMonitor** component file (`.xJobMonitor`) and double-click it.

   The **JobMonitor** utility opens on a new tab in the **Development** window.

2. In the **File Systems** window, locate the file for the **event** or **process flow** you want to monitor then drag and drop the file onto the **Job Monitor** screen in the **Development** window.

   Note: For more information on events and how to configure them, see "Events" on page 158. For more information on process flows and how to configure them, see "Process Flows" on page 63.

   The **Add Monitor** dialog displays.

3. On the **Alert** tab of the Add Monitor dialog, enter a name for your alert in the **Name** field.

   Note: The **Enabled** check box is selected by default when you are configuring a new alert, but you can clear it if you need to temporarily

deactivate the alert. On the main JobMonitor screen, you can quickly enable and disable your alerts with the alert's respective Enable check box in the JobMonitor table.

4. Depending on which type you are adding to the JobMonitor, either the **Service Configuration** or **Process Flow Configuration** field displays the file path and name of your chosen event or process flow.

   If you no longer want to monitor the selected configuration, you may change your selection by clicking the **Ellipsis** button associated with the configuration type and from the dialog that appears, selecting another configuration to monitor.

5. Optionally, you may enter a brief explanation of the alert's purpose in the **Description** field. This description is viewable in the table on the main JobMonitor screen to help you distinguish between your alerts.

   When you are finished, select the **Alert Flow** tab.

6. You must select the action that the JobMonitor takes when an alert arises. Next to the **Process Flow to run when an alert occurs** field, click the **Ellipsis** button and the **Choose a Process Flow** dialog displays. Select the Process Flow to run and click **OK**.

   The **Alert Flow** tab reappears.

7. Any variables needed for the alert can be mapped or created at this time. These variables define the details that can be included and used within the process flow that gets kicked off when an alert is triggered.

   If you are configuring **Variables created by the JobMonitor**, the variables are listed in the **Name** column and you can double-click in the respective cell in the **Map To** column in order to add a value. The following variables are created by the JobMonitor:

   - **ConfigFile.** Indicates the configuration file for the event or process flow being monitored.
   - **ClusterNode.** Indicates the node on which the monitored component last ran.
   - **LastRun.** Indicates the time (string – timestamp) when the component last ran.
   - **LastTest.** Indicates the time (string – timestamp) of the last test/query.

   If you are creating **User defined variables**, click the **Add** button to insert a new row in the table and double-click on the appropriate cells in the **Name** and **Value** columns to add values for your custom variables.

   When you are finished, select the **Schedule** tab.

8. On the **Schedule** tab, you must set the times for which the JobMonitor will be actively observing whether or not your selected event or process flow encounters any errors.

Using the **Days of Week** check boxes, select the days you want the JobMonitor running the verification mechanism.

9. With the **Time** fields, indicate the time to **Start** and **Stop** the JobMonitor scans.

10. In the **Interval** fields, you must specify the time interval between JobMonitor tests which check if an event or process flow has launched. You can select from **second**, **minute**, or **hour** measurements for the intervals.

    For example, in the schedule below, the JobMonitor will run checks between midnight Saturday morning and 11:59PM Sunday night. Then if the configured event or process flow hasn't initiated anything within the last 2 minutes, the error process flow will launch.

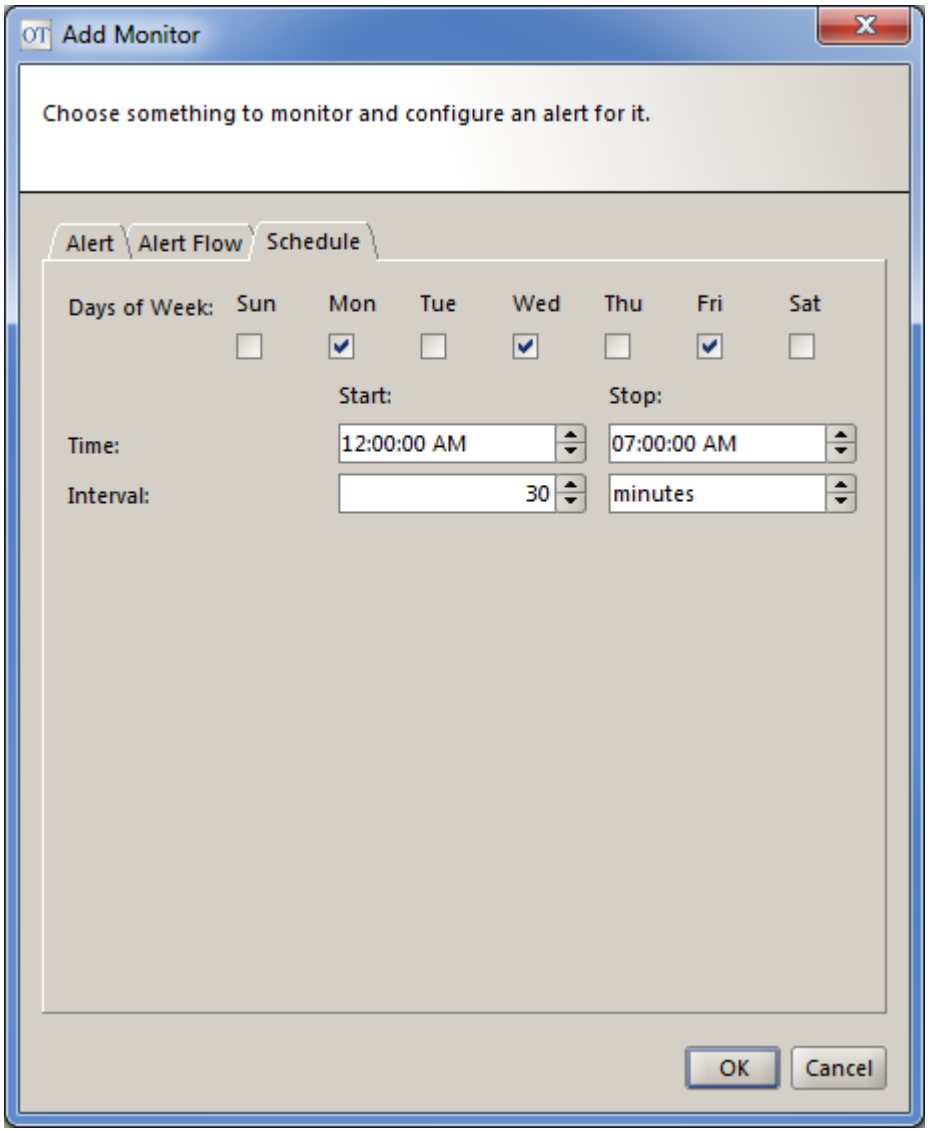


Figure 5-15: Add Monitor – Schedule tab

When you are finished setting the schedule, click **OK**.

The **JobMonitor** screen appears with your new alert in the table.

11. Make sure that you turn on the alert(s) you want to use by selecting the check box in the **Enabled** column.

    Once you **save** your JobMonitor configuration, it is ready for use.

5.6.6 MailService

The MailService component handles the sending and receiving of email messages.

MailService Properties

The following parameters are set for the MailService component:

- **MailSender**. Sends email messages to a mail server.
  - **Enabled**. Specifies whether the MailSender parameter is enabled or not.
  - **MailServer**. Details the MailServer configuration settings.
    - **Host**. Identifies the name of the server with which to connect.
    - **Port**. Identifies the port on the server you wish to connect to.
    - **User**. Specifies the login user ID to use.
    - **Password**. Specifies the login password that corresponds to the above user ID.
    - **Timeout**. Indicates the amount of time, in milliseconds, before the session will expire.
  - **SendProtocol**. Identifies the name of the email protocol used to send messages to a mail server. Options are **SMTP** or **IMAP4**.
  - **AuthenticationMode**. Specifies the mail server authentication mode to use. Options are **NO_AUTHENTICATION**, **STANDARD** or **SECURE**.
  - **Folder**. Displays the name of the folder where emails are stored. This parameter is used when the email protocol type is IMAP4.
  - **ReliableTransfer**. Contains the settings for the ReliableTransfer mechanism that is activated when an error occurs during the transmission of data. For more information, see "ReliableTransfer" on page 278 .
    - **Retries**. Specifies the number of times to retry transferring data when an error occurs. When this parameter is set to 0, no retries will be attempted and ReliableTransfer is not activated.
    - **TimeBetweenRetries**. Specifies the amount of time, in milliseconds, to wait between retries.
    - **OnError**. Specifies how to handle error conditions once all retries have been exhausted and an error still persists. If you select **Log**, the error will be logged and the process will appear to be successful to the enclosing process flow. If you select **ThrowException**, and exception will be thrown to the enclosing process flow.

5.6.7 MessageService

The MessageService handles the sending and resending of messages and acknowledgements.

5.6.7.1 MessageService Properties

The following parameters are set for the MessageService:

- **SendFlow**. Specifies the process flow to run when sending a message.
- **AckFlow**. Specifies the process flow to run when an acknowledgement is received.
- **FailedFlow**. Specifies the process flow to run when a message has not received an acknowledgement, and the number of times tried has exceeded the maximum.
- **MinutesBetweenAttempts**. Specifies the number of minutes to wait between attempts to resend the message.
- **NumberOfRetries**. Indicates the number of attempts to resend the message. If a negative value is entered, the message will be sent until an acknowledgement is received.
- **HolderDirectory**. Specifies the directory where all incoming files are held.
- **InputMapName**. Specifies the name of the input map used to store the data to be processed. This name is used as the key in the input map, which can be used later to retrieve and map the data.

5.6.8 MshService

The **MshService** (Message Service Handler) functions as a service component and listens to incoming ebXML messages and then submits a process flow.

For more information, see *OpenText Output Transformation Server ebXML Messaging - User Guide (VDTOTS-H-UXM)*.

5.6.9 Timer

The **Timer** service component implements a timekeeping mechanism to record the length of time certain portions of code take to complete. This information is then written to a log.

5.6.9.1 Timer Properties

The following properties are available:

- **ResultsFile**. Specifies the path to the Timer log. Each time the service is restarted the log is overwritten.
- **ToSystemOut**. Indicates that timer statistics will also be written to the system.out file. By default, this is **off**.

5.7 System Components

Topics in this section discuss system components.

5.7.1 Custom Components

If you require additional or customized functionality that is not provided by Output Transformation Server standard components, such as custom routing or integration with a back office system, you can create custom components. The **Custom Component Wizard** guides you through the steps for creating a skeleton Java class that you can fill in with your own custom business logic.

The process for creating and implementing custom components is documented in *OpenText Output Transformation Server - Developer's Guide (VDTOTS-H-PGD)*.

5.7.2 Logging Profile

Logging profiles are used by the system to specify what events to log. Log profiles are turned on and off by adding or removing them from the System Configuration file.

When creating components for process flows, you may wish to alter the default logging profile for your project by creating a Logging Profile component and adding it to a Default System profile.

- "System Configuration" on page 17
- "System Configuration" on page 263
- "Logging Profiles" on page 46

5.7.3 System Configuration

As part of the installation, there is a default configuration file loaded. If you want to define a new default system configuration in a specific process flow, create a System Configuration component to be used in the process flow.

For more information, see "System Configuration Parameters" on page 17.

5.8 Tool Components

This section describes the LoadSimulator tool component.

5.8.1 LoadSimulator

The **LoadSimulator** component is a utility that can repeatedly run one or more job and record the performance statistics in order to simulate the load that the server(s) would encounter in real-world conditions. The tool can be configured to run with Output Transformation Server and Output Transformation Engine features.

5.8.1.1 LoadSimulator Component

The **LoadSimulator** component is a utility that can repeatedly run one or more job and record the performance statistics in order to simulate the load that the server(s) would encounter in real-world conditions. The tool can be configured to run with Output Transformation Server and Output Transformation Engine features.

Initially testing jobs with the LoadSimulator is an indispensable resource for:

- Establishing that a particular server can handle a desired throughput, or alternatively, verifying what a server's throughput is.
- Confirming that a load can be sustained over an extended period of time (days/weeks).
- Detecting the amount of memory required in order to maintain processing of a load.
- Providing simulated runs on the server that can be monitored throughout with Task Manager or other similar performance utilities.

LoadSimulator Properties

Some parameters may require different types of values, depending on which OpenText products you are working with.

- **VisionSetup**. Defines configuration properties on how Output Transformation Engine starts and jobs are run.
  - **D2esys**. Specifies the Output Transformation Engine system configuration (d2esys) used to start Output Transformation Engine.

- **Rasterize**. Indicates whether Output Transformation Engine jobs should rasterize all text.
- **RespectInputList**. Signifies whether the use of .inputList files are recognized by the project. These input list files contain one or more lines of input file patterns that route to input files to use with the given Output Transformation Engine project. For example, when the pdf2tif.d2eproj is used and RespectInputList is enabled, `pdf2tif.d2eproj.inputList` is checked. If it does not exist, the LoadSimulator automatically looks for a `Global.inputList` file. As input list files are read, each line is handled as a pattern relative (downwards) from the location of the original Output Transformation Engine project file. Therefore, if the input list contained `*.pdf`, then all `.pdf` files in the directory are processed with the `pdf2tif.d2eproj`. On the contrary, if this parameter is set to false or if neither inputList files exist, the Output Transformation Engine project is run without any overriding input and only the input specified by the Output Transformation Engine project is used.
- **FdOutputMode**. Determines the action to perform with the output. There are four options:
  - **THROWAWAY** – All writing to fdOutput uses a DummyOutputStream that calculates the bytes written, but discards the actual output.
  - **PIPE_TO_FILE** – Instead of writing to fdOutput, the output is sent to an output file as indicated in the BaseOutputPath parameter.
  - **NO_ACTION** – The fdOutput is written according to the specifications in your Output Transformation Engine project.
  - **SET_VARS** – Automatically assigns numerous job variables to allow the project to use the variables in fdOutput. (Currently reserved for future use.)

- **EsSetup**. Defines configuration properties on how Output Transformation Server jobs are run.
  - **EsMode**. Indicates the method of how Output Transformation Server jobs are submitted. The following options are available:
    - **Direct** – Output Transformation Server is run on the same Java Virtual Machine (JVM). If Output Transformation Server is not currently running, it will be started when the job is initialized and then shut down again when the LoadSimulator is finished running.
    - **RestWebService** – Output Transformation Server is run through the REST JobRunner API. When using the RestWebService setting, if the ContextRoot parameter is left empty then the value is automatically substituted with the `rest/api/v1` value. Also, if you are running jobs from the server, you must have the same system configuration settings on your local machine as the server in order to properly filter out jobs that are ready to run.

  - **Host**. Specifies the IP address or host name of the Output Transformation Server server. If you are connecting to servers running as a cluster, only enter the IP address of your main connection point.
  - **ContextRoot**. Denotes the context root used to set up the Output Transformation Server EAR.
  - **Port**. Specifies the port number used to set up the Output Transformation Server EAR.
  - **ResultMode**. Determines the action to perform with the Output Transformation Server output. There are two options:
    - **THROWAWAY** – All results are discarded.
    - **WRITE_TO_FILE** – All results are written to disk. (Currently reserved for future use.)
  - **JobVarResultFilename**. Indicates where job variables returned from the Output Transformation Server job are written to. (Currently reserved for future use.)
- **RepoSetup**. Defines configuration properties on how the repository services behave when search and retrieval requests are submitted.
  - **RepoWsdl**. Specifies the URL for the repository services WSDL. For example, http://127.000.00.255:8080/repo/services/RepositoryServices?wsdl.
  - **RepoApi**. Specifies the URL for the repository API. For example, http://api.repository.company.com.
  - **RetrieveMode**. Determines the action to perform with the retrieval results. There are two options:
    - **THROWAWAY** – All results are discarded.
    - **WRITE_TO_FILE** – All results are written to disk. (Currently reserved for future use.)
- **PreloadLimitK**. Sets the maximum file size in kilobytes that can be preloaded into memory. Preloading a file into memory permits jobs to be submitted without the need to individually load the input from disk each time. A value of 0 indicates that no files are preloaded, while -1 means that there is no limit. The default setting is **-1**.
- **StatusDelaySeconds**. Determines the number of seconds between the LoadSimulator's ThreadPool and Thread status messages.
- **PauseAtVeryEnd**. Indicates whether all CPU activity should be temporarily halted when the LoadSimulator is finished running. This intermission allows monitoring tools such as Task Manager to gather statistics or a Java Profiler snapshot to be captured.
- **TimestampLogs**. Designates whether log files should be appended with a time stamp, making each entry unique. This is useful when you are running multiple times and comparing the results of the runs.

- **BaseOutputPath**. Specifies the file path location where output files/logs from the LoadSimulator are written. This location value is relative to the location of the config file.
- **Worksets**. Contains configuration settings for the one or more items of work to process, including details on which projects to run and if Output Transformation Engine or Output Transformation Server are used.
  - **Name**. Specifies the name used to identify this particular workset instance.
  - **IsActive**. Denotes whether this particular workset instance is used. This allows users to set up multiple worksets at a time, enabling and disabling individual worksets as they are required.
  - **RunMode**. Indicates the application that is running: **ES** for Enterprise Server (Output Transformation Server), **Vision** for d2e Vision (Output Transformation Engine), or **REPO** for ES Repository.
  - **PauseBefore**. Indicates whether all CPU activity should be temporarily halted before a project begins to run. This intermission allows you to set up any additional monitoring tools. You need to press **Enter** in order to start the workset.
  - **PauseAfter**. Indicates whether all CPU activity should be temporarily halted after the workset is finished running. You need to press **Enter** in order to end the workset and have the thread pool continue with any remaining worksets.
  - **ThreadPoolName**. Designates the name of the thread pool. A thread pool can only be used on one workset at a time; therefore, worksets having the same thread pool name are run consecutively. If you want multiple worksets to run concurrently, you must use a different thread pool name for each workset.
  - **ConcurrentJobs**. Determines the number of jobs to run simultaneously.
  - **RunTimeMinutes**. Identifies the length of time in minutes to run the simulation. If you do not want to impose a limit on the time, use **-1** for no limit. However, both this and the RunCount parameter cannot be set to -1.
  - **RunCount**. Identifies the number of times to run each project and input combination. If you do not want to impose a limit on the number of runs, use **-1** for no limit. However, both this and the RunTimeMinutes parameter cannot be set to -1.
  - **DelayTimeMinutes**. Specifies the length of time in minutes to wait before starting this workset.
  - **BetweenJobInterval**. Contains configuration settings on the delay time between initializing new jobs.
    - **MinWaitTimeMillis**. Identifies the minimum length of time in milliseconds to wait.
    - **MaxWaitTimeMillis**. Identifies the maximum length of time in milliseconds to wait. A different random number between your designated minimum and maximum wait time is used each time a job is run, which determines the waiting period.

    - **IsInclusive**. Indicates whether the designated wait time includes the time spent running the previous job. (Currently reserved for future use.)
  - **ProjectBasePath**. Details the base file path location of your projects. This value can be either an absolute or relative file path. When entering a relative value, the path is relative to the location specified in the BasePath parameter, or the location of the parm object in standalone mode.
  - **ProjectToRun**. Designates the project name, or for multiple projects the project pattern, to run. The array syntax allows for the asterisk wildcard character, permitting values such as `*.d2eproj`, `*afp2pdf.d2eproj`, or `**/*.d2eproj` to be valid.
  - **ProjectFileList**. Specifies the location of a file containing a list of projects to run. If this parameter is used, it overrides any value in the ProjectToRun parameter.
  - **OutputExt**. Designates the extension to use when writing output.
  - **Jobvars**. Contains a list of job variables and values to pass into jobs.
    - **Name**. Indicates the job variable's name.
    - **Value**. Indicates the job variable's value.
  - **Inputs**. Denotes a list of input names and sources for Output Transformation Server when defining multiple input sources.
    - **Name**. Specifies the input list name. This parameter applies to Output Transformation Server only.
    - **Filenames**. Indicates the path and file names relative to the project file. If this parameter has no value, the project's inputList file is used. To send a multi-input XDoc, use more than one Filenames parameter.
  - **Results**. Denotes a list of result names and targets for Output Transformation Server when defining multiple results.
    - **Name**. Specifies the result name. This parameter applies to Output Transformation Server only.
    - **Ext**. Designates the file extension to use when writing the result.
- **BasePath**. Identifies the home directory containing project files and logs folders to use when running from Output Transformation Server.
- **CsvXDoc**. Indicates the name of an XDoc containing CSV-formatted results to create. If left blank, CSV results are written to the `basepath\logs\Simulator_CSV.txt` file.
- **TabDelimit**. Indicates whether the CSV file is delimited. If the value is false, the CSV file will contain multi-spaced columns.

5.8.1.2 LoadSimulator Logs

A series of log files are generated by the LoadSimulator component that provide various performance, status, and property statistics. The location where log files are written is indicated in the BaseOutputPath parameter. These log files are created:

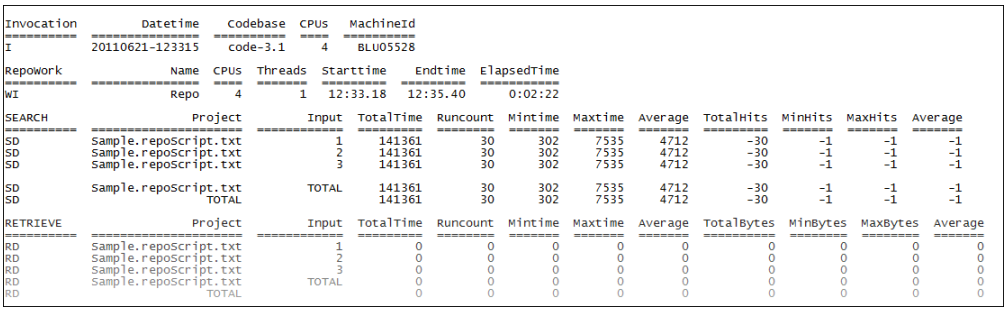
Log File Name	Description
LoadSimulator_Detail.log	Provides details of the jobs being submitted. When running within Output Transformation Server, these are written as DEBUG messages.
LoadSimulator_Summary.log	Provides a summary of the work completed. When running within Output Transformation Server, these are written as INFO messages.
LoadSimulator_Status.log	Provides the current status of each thread pool and its threads. When running within Output Transformation Server, these are written as INFO messages.
LoadSimulator_Warning.log	Provides any warning messages issued by the jobs.
LoadSimulator_CSV.log	Provides a broad summary of information in a CSV-format file, which can be imported into Microsoft Excel.

5.8.1.3 Reading the LoadSimulator CSV Log

The LoadSimulator component generates a `LoadSimulator_CSV.log` file which provides an all-encompassing summary of such statistics as project names, inputs used, file sizes, and run times, among others. Also, if you prefer, the CSV file can be separately imported into Microsoft Excel.

The location of this file is specified by the LoadSimulator component's **BasePath** parameter. This parameter specifies the home directory location where all project files and logs are stored. Some other parameters allow you to modify the CSV file's delivery method. With the **CsvXDoc** parameter, you can opt to create an XDoc containing CSV-formatted results by entering an XDoc name, otherwise the CSV results are written to the `basepath\logs\Simulator_CSV.txt` file. Moreover, you can opt for a delimited CSV file by setting the **TabDelimit** parameter to true; if the value is false, the generated CSV file contains multi-spaced columns.

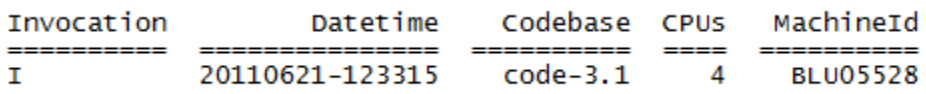
Understanding how to read the LoadSimulator's CSV log file is imperative to the component's use as you will be able to make better-informed decisions about your current configuration setup.



```
Invocation          Datetime     Codebase  CPUs   MachineId
==========  ================  ==========  ====  ==========
I           20110621-123315     code-3.1     4     BLU05528

RepoWork               Name  CPUs  Threads  Starttime    Endtime  ElapsedTime
==========  ================  ====  =======  =========  =========  ===========
WI                     Repo     4        1   12:33.18   12:35.40      0:02:22

SEARCH                  Project         Input  TotalTime  Runcount  Mintime  Maxtime  Average  TotalHits  MinHits  MaxHits  Average
==========  ====================  ===========  =========  ========  =======  =======  =======  =========  =======  =======  =======
SD          Sample.reposcript.txt            1     141361        30      302     7535     4712        -30       -1       -1       -1
SD          Sample.reposcript.txt            2     141361        30      302     7535     4712        -30       -1       -1       -1
SD          Sample.reposcript.txt            3     141361        30      302     7535     4712        -30       -1       -1       -1

SD          Sample.reposcript.txt        TOTAL     141361        30      302     7535     4712        -30       -1       -1       -1
SD                           TOTAL               141361        30      302     7535     4712        -30       -1       -1       -1

RETRIEVE                Project         Input  TotalTime  Runcount  Mintime  Maxtime  Average  TotalBytes  MinBytes  MaxBytes  Average
==========  ====================  ===========  =========  ========  =======  =======  =======  ==========  ========  ========  =======
RD          Sample.reposcript.txt            1          0         0        0        0        0           0         0         0        0
RD          Sample.reposcript.txt            2          0         0        0        0        0           0         0         0        0
RD          Sample.reposcript.txt            3          0         0        0        0        0           0         0         0        0
RD          Sample.reposcript.txt        TOTAL          0         0        0        0        0           0         0         0        0
RD                           TOTAL                     0         0        0        0        0           0         0         0        0
```

Figure 5-16: Sample Snippet of the LoadSimulator CSV Log

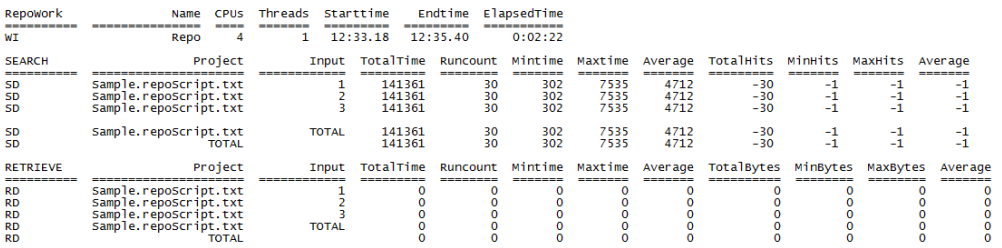
The ===== lines are simply dividers and indicate the separate columns, while the text above each is the title of the column. Moreover, the first column describes the type of information listed in the subsequent detail lines. The following statistics are shown in the log according to the row, from left to right.



```
Invocation          Datetime     Codebase  CPUs   MachineId
==========  ================  ==========  ====  ==========
I           20110621-123315     code-3.1     4     BLU05528
```

Figure 5-17: LoadSimulator CSV Log - Invocation Section

Column	Description
Invocation	Provides details on this invocation of the simulator.
Datetime	Indicates the date and time that the simulation began running in YYYYMMDD-HHMMSS format.
Codebase	Designates the version of the code used to run the project.
CPUs	Specifies the total number of available CPUs used during the simulated run. Using the LoadSimulator component's **PauseBefore** or **PauseAfter** parameters, you can adjust the number of available CPUs using the **Set Affinity** feature in Windows Task Manager to test various workset loads. Accordingly, repeated worksets with different numbers of CPUs available can be performed under the same invocation.
MachineId	Reveals the machine identification name.

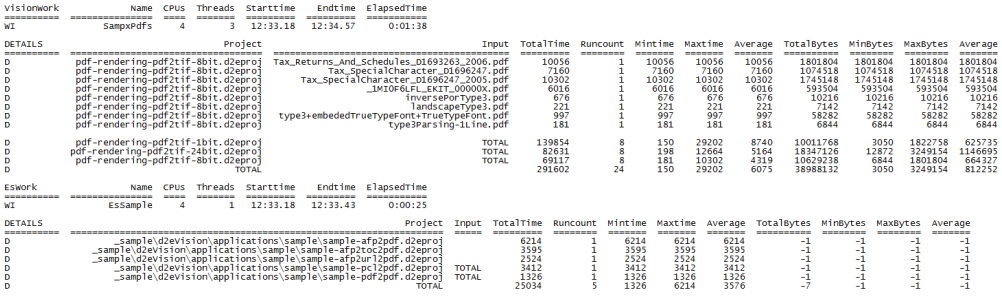


```
RepoWork              Name  CPUs  Threads  Starttime    Endtime  ElapsedTime
==========  ==============  ====  =======  =========  =========  ===========
WI                  Repo     4        1   12:33.18   12:35.40       0:02:22

SEARCH                   Project         Input  TotalTime  Runcount  Mintime  Maxtime  Average  TotalHits  MinHits  MaxHits  Average
==========  ====================  ============  =========  ========  =======  =======  =======  =========  =======  =======  =======
SD          Sample.reposcript.txt             1     141361        30      302     7535     4712        -30       -1       -1       -1
SD          Sample.reposcript.txt             2     141361        30      302     7535     4712        -30       -1       -1       -1
SD          Sample.reposcript.txt             3     141361        30      302     7535     4712        -30       -1       -1       -1

SD          Sample.reposcript.txt         TOTAL     141361        30      302     7535     4712        -30       -1       -1       -1
SD                          TOTAL               141361        30      302     7535     4712        -30       -1       -1       -1

RETRIEVE                 Project         Input  TotalTime  Runcount  Mintime  Maxtime  Average  TotalBytes  MinBytes  MaxBytes  Average
==========  ====================  ============  =========  ========  =======  =======  =======  ==========  ========  ========  =======
RD          Sample.reposcript.txt             1          0         0        0        0        0           0         0         0        0
RD          Sample.reposcript.txt             2          0         0        0        0        0           0         0         0        0
RD          Sample.reposcript.txt             3          0         0        0        0        0           0         0         0        0
RD          Sample.reposcript.txt         TOTAL          0         0        0        0        0           0         0         0        0
RD                          TOTAL                        0         0        0        0        0           0         0         0        0
```

Figure 5-18: LoadSimulator CSV Log - RepoWork Section

Column	Description
RepoWork	Provides details on each project/input combination, total for each project (if more than one input), and the total for all projects. These details are split into **Search** and **Retrieve** sections.
Name	Indicates the name of the workset.
CPUs	Specifies the total number of available CPUs used during the simulated run. Using the LoadSimulator component's **PauseBefore** or **PauseAfter** parameters, you can adjust the number of available CPUs using the **Set Affinity** feature in Windows Task Manager to test various workset loads. Accordingly, repeated worksets with different numbers of CPUs available can be performed under the same invocation.
Threads	Designates the number of threads used during the simulated run.
Starttime	Indicates the time which the job was initiated.
Endtime	Indicates the time which the job was completed.
ElapsedTime	Declares the total time that passed in HH:MM:SS format.
Search	Describes the search request particulars.
Project	Denotes the name of the repository script that was run. This script specifies the search or optional retrieve request to run.
Input	Specifies the individual run number from within the script.
TotalTime	Expresses the total time in seconds that elapsed for the individual project/input search or retrieve combination.

Runcount	Specifies the total number of times the project was run.
Mintime	Designates the fastest amount of time in seconds that it took to complete a single run of the indicated project.
Maxtime	Designates the longest amount of time in seconds that it took to complete a single run of the indicated project.
Average (Time)	Displays the average amount of time in seconds it took to complete a single run of the indicated project.
TotalHits	Indicates the total number of matching results in the chosen project for your search criteria.
MinHits	Signifies the lowest number of matching results in the chosen project for your search criteria.
MaxHits	Signifies the highest number of matching results in the chosen project for your search criteria.
Average (Hits)	Displays the average number of matching results in the chosen project for your search criteria.
Retrieve	Describes the retrieve request particulars.
TotalBytes	Indicates the total number of transferred bytes received during your project run.
MinBytes	Declares the lowest number of transferred bytes received during your project run.
MaxBytes	Declares the highest number of transferred bytes received during your project run.
Average (Bytes)	Displays the average number of transferred bytes received during your project run.



```
VisionWork              Name  CPUs  Threads  Starttime    Endtime  ElapsedTime
==========  ==============  ====  =======  =========  ========  ===========
WI                SampxPdfs     4        3   12:33.18  12:34.57       0:01:38

DETAILS                                 Project                                                 Input  TotalTime  Runcount  Mintime  Maxtime  Average  TotalBytes  MinBytes  MaxBytes  Average
==========  ==========================================  ==================================================  =========  ========  =======  =======  =======  ==========  ========  ========  =======
D           pdf-rendering-pdf2tif-8bit.d2eproj  Tax_Returns_and_Schedules_D1693263_2006.pdf      10056         1    10056    10056    10056     1801804   1801804   1801804  1801804
D           pdf-rendering-pdf2tif-8bit.d2eproj                Tax_SpecialCharacter_D1696247.pdf       7160         1     7160     7160     7160     1074518   1074518   1074518  1074518
D           pdf-rendering-pdf2tif-8bit.d2eproj       Tax_SpecialCharacter_D1696247_2005.pdf      10302         1    10302    10302    10302     1745148   1745148   1745148  1745148
D           pdf-rendering-pdf2tif-8bit.d2eproj                  _IMIOF6LFL_EXIT_00000X.pdf       6016         1     6016     6016     6016      593504    593504    593504   593504
D           pdf-rendering-pdf2tif-8bit.d2eproj                          inversePorType3.pdf        676         1      676      676      676       10216     10216     10216    10216
D           pdf-rendering-pdf2tif-8bit.d2eproj                          landscapeType3.pdf        221         1      221      221      221        7142      7142      7142     7142
D           pdf-rendering-pdf2tif-8bit.d2eproj  type3+embededTrueTypeFont+TrueTypeFont.pdf        997         1      997      997      997       58282     58282     58282    58282
D           pdf-rendering-pdf2tif-8bit.d2eproj                      type3Parsing-1Line.pdf        181         1      181      181      181        6844      6844      6844     6844

D           pdf-rendering-pdf2tif-1bit.d2eproj                                       TOTAL     139854         8      150    29202     8740    10011768      3050   1822758   625735
D          pdf-rendering-pdf2tif-24bit.d2eproj                                       TOTAL      82631         8      198    12664     5164    18347126     12872   3249154  1146695
D           pdf-rendering-pdf2tif-8bit.d2eproj                                       TOTAL      69117         8      181    10302     4319    10629238      6844   1801804   664327
D                                       TOTAL                                                 291602        24      150    29202     6075    38988132      3050   3249154   812252

EsWork                  Name  CPUs  Threads  Starttime    Endtime  ElapsedTime
==========  ==============  ====  =======  =========  ========  ===========
WI                EsSample     4        1   12:33.18  12:33.43       0:00:25

DETAILS                                                                                 Project  Input  TotalTime  Runcount  Mintime  Maxtime  Average  TotalBytes  MinBytes  MaxBytes  Average
==========  ==================================================================================  =====  =========  ========  =======  =======  =======  ==========  ========  ========  =======
D                                  _sample\d2evision\applications\sample\sample-afp2pdf.d2eproj           6214         1     6214     6214     6214          -1        -1        -1       -1
D                              _sample\d2evision\applications\sample\sample-afp2toc2pdf.d2eproj           3595         1     3595     3595     3595          -1        -1        -1       -1
D                              _sample\d2evision\applications\sample\sample-afp2ur12pdf.d2eproj           2524         1     2524     2524     2524          -1        -1        -1       -1
D                                  _sample\d2evision\applications\sample\sample-pcl2pdf.d2eproj  TOTAL     3412         1     3412     3412     3412          -1        -1        -1       -1
D                                  _sample\d2evision\applications\sample\sample-pdf2pdf.d2eproj  TOTAL     1326         1     1326     1326     1326          -1        -1        -1       -1
D                                                                                        TOTAL    25034         5     1326     6214     3576          -7        -1        -1       -1
```

Figure 5-19: LoadSimulator CSV Log - VisionWork and EsWork Sections

Column	Description
VisionWork/EsWork	Provides details on each project/input combination, total for each project (if more than one input), and the total for all projects.
Name	Indicates the name of the workset.
CPUs	Specifies the total number of available CPUs used during the simulated run. Using the LoadSimulator component's **PauseBefore** or **PauseAfter** parameters, you can adjust the number of available CPUs using the **Set Affinity** feature in Windows Task Manager to test various workset loads. Accordingly, repeated worksets with different numbers of CPUs available can be performed under the same invocation.
Threads	Designates the number of threads used during the simulated run.
Starttime	Indicates the time which the job was initiated.
Endtime	Indicates the time which the job was completed.
ElapsedTime	Declares the total time that passed in HH:MM:SS format.
Details	Describes the project particulars.
Project	Denotes the name of the project that was run.
Input	Specifies the number in the sequence that the project was run.
TotalTime	Expresses the total time in seconds that elapsed for the individual job.
Runcount	Specifies the total number of times the project was run.
Mintime	Designates the fastest amount of time in seconds that it took to complete a single run of the indicated project.
Maxtime	Designates the longest amount of time in seconds that it took to complete a single run of the indicated project.
Average (Time)	Displays the average amount of time in seconds it took to complete a single run of the indicated project.
TotalBytes	Indicates the total number of transferred bytes received during your project run.

MinBytes	Declares the lowest number of transferred bytes received during your project run.
MaxBytes	Declares the highest number of transferred bytes received during your project run.
Average (Bytes)	Displays the average number of transferred bytes received during your project run.

5.9 Other Components

The topics in this section describe the other components available for use in OpenText Output Transformation Server:

5.9.1 CPA

The CPA (Collaboration Protocol Agreement) component defines the details of the transport, messaging, security, reliability, and synchronization mode that enable business documents to be interchanged between parties in business collaboration.

CPA is part of OpenText Output Transformation Server ebXML Messaging and is documented in *OpenText Output Transformation Server ebXML Messaging User Guide*.

5.9.2 Parm Definition

The **Parm Definition** component is used to build or modify parameter definitions for parm objects to add user properties to custom components.

For more information, see *OpenText Output Transformation Server - Developer's Guide (VDTOTS-H-PGD)*

5.9.3 Text File

There are no parameters for the text file; it simply contains text that you add. Create the component and type the desired text in the editable document in the Development window.

5.9.4 XML Document

An XML document component can be added to process flows.

There are no parameters for the XML file; it simply contains a properly formatted XML document that you add. Create the component and enter the desired XML document in the editable document in the Development window.

5.10 Functions

A **function** is a segment of Java code written to perform a calculation upon, or manipulation of, input data. It is used to produce output results by specifying that function within an output item value.

Functions can be added or configured:

- In the **Function Editor** of the JobVariableRouter.
- In any service, event, and process parameter that has a text field. For example, the FileProcess parameter, **FileName**, can access the **sysDate** function to insert the system date in the output file's name:

```
<install_home>\testResult\@sysDate("yyyyMMdd")-FileResult.xml
```

The predefined functions available in Output Transformation Server are grouped into the following categories: date, numeric, object, and string functions.

Related Links

- "JobVariableRouter" on page 201
- "Adding a Function" on page 202
- "FileProcess" on page 198

5.10.1 Date Functions

The following date functions are available in Output Transformation Server:

Function	Parameters and Return Types
dayOfMonth. Returns a number representing the day portion of the given date. The value returned is between 1 and 31.	• date (type=java.lang.Object) • pattern (type=java.lang.String)
dayOfWeek. Returns a number representing the day of the week in which the given date resides. The value returned is between 1 and 7.	• date (type=java.lang.Object) • pattern (type=java.lang.String)
endOfMonth. Returns a number representing the last day of the month in which the given date resides. The value returned is between 28 and 31.	• date (type=java.lang.Object) • pattern (type=java.lang.String)
month. Returns a number representing the month portion of the given date. The value returned is between 0 and 11, with 0 representing January.	• date (type=java.lang.Object) • pattern (type=java.lang.String)
sysDate. Returns the system date as a string in the specified format. If the parameter is omitted, the format is MM/DD/YYYY.	• date (type=java.lang.String)

Function	Parameters and Return Types
time. Returns the current time as the number of milliseconds since January 1, 1970, 00:00:00 GMT. This value is a string.	No parameters
year. Returns a number representing the year portion of the given date.	• date (type=java.lang.Object) • pattern (type=java.lang.String)

5.10.2 Numeric Functions

The following numeric functions are available in Output Transformation Server:

Function	Parameters and Return Types
add. Adds two numbers to return a number as a string.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
avg. Returns a string which is the average of input numerical values.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
compare. Compares the first numeric parameter with the second one. If the first parameter is less than the second one, then function returns -1. If the first parameter is greater than the second one, then function returns 1. If the parameters are equal, then function returns 0.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
divide. Divides two numbers to return a result as a string.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
equalTo. Compares the first numeric parameter with the second one. If the first parameter is equal to the second one, then function returns True; otherwise False.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
greaterThan. Compares the first numeric parameter with the second one. If the first parameter is greater than the second one, then function returns True; otherwise False.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
lessThan. Compares the first numeric parameter with the second one. If the first parameter is less than the second one, then function returns True; otherwise False.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
max. Returns a string which is the bigger number among input numerical values.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
min. Returns a string which is the smaller number among input numerical values.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)

Function	Parameters and Return Types
mod. Returns a string that results from performing a modulus operation on parameter one by parameter two.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
multiply. Multiplies two numbers to return a result as a string.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)
round. Returns rounded number as a string.	• number (type=java.lang.Double)
subtract. Subtracts second number from the first one to return a result as a string.	• number1 (type=java.lang.Double) • number2 (type=java.lang.Double)

5.10.3 Object Functions

The following object functions are available in Output Transformation Server:

Function	Parameters and Return Types
isNull. Checks whether the value of the input parameter is null or not.	• object (type=java.lang.Object)
sizeOfInputXDoc. Returns the size of the input XDoc in bytes.	• xdocName (type=java.lang.String)
sizeOfResultXDoc. Returns the size of the result XDoc in bytes.	• xdocName (type=java.lang.String)

5.10.4 String Functions

The following string functions are available in Output Transformation Server:

Function	Parameters and Return Types
concat. Concatenates two input strings.	• string1 (type=java.lang.String) • string2 (type=java.lang.String)
deleteStr. Deletes characters from the original string. If the substring that is to be deleted is not found in the input, the entire input string is returned.	• string (type=java.lang.String) • beginIndex (type=java.lang.Integer) • length (type=java.lang.Integer
endsWith. Tests if the first string ends with the suffix specified by the second string. Returns True if the second argument is a suffix of the first one; otherwise returns False.	• string1 (type=java.lang.String) • string2 (type=java.lang.String)
equals. Compares the first string to the second string.	• string1 (type=java.lang.String) • string2 (type=java.lang.String)
fill. Repeats an input string or expression 'n' times.	• string1 (type=java.lang.String) • count (type=java.lang.Integer)

Function	Parameters and Return Types
flip. Reverses an input string to its mirror image.	• string (type=java.lang.String)
indexOf. Returns the index within the first string of the first occurrence of the specified second string, starting at the specified index.	• string1 (type=java.lang.String) • string2 (type=java.lang.String) • fromIndex (type=java.lang.Integer)
insertStr. Inserts one string into another. If the substring that is to be inserted is not found in the input, the entire input string is returned.	• target (type=java.lang.String) • source (type=java.lang.String) • beginIndex (type=java.lang.Integer) • length (type=java.lang.Integer)
lastIndexOf. Returns the index within the first string of the rightmost occurrence of the specified second string.	• string1 (type=java.lang.String) • string2 (type=java.lang.String) • fromIndex (type=java.lang.Integer)
length. Returns the length of the string.	• string1 (type=java.lang.String)
startsWith. Tests if the first string starts with the prefix specified by the second string. Returns True if the second argument is a prefix of the first one; otherwise returns False.	• string1 (type=java.lang.String) • string2 (type=java.lang.String)
subString. Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex -1. Thus the length of the substring is endIndex - beginIndex.	• string (type=java.lang.String) • beginIndex (type=java.lang.Integer) • endIndex (type=java.lang.Integer)
toLowerCase. Converts all of the characters in the string to lower case using the rules of the default locale.	• string (type=java.lang.String)
toUpperCase. Converts all of the characters in the string to upper case using the rules of the default locale.	• string (type=java.lang.String)
trim. Returns a copy of the string with leading and trailing white spaces omitted.	• string (type=java.lang.String)

5.11 ReliableTransfer

ReliableTransfer is a mechanism that allows the transfer of data to automatically restart or resume if an error occurs during the transfer. If the transfer fails while a process is sending outbound data, and ReliableTransfer is activated, the process will sleep for a specified time. When the interval is elapsed, the process retries to transfer data again. This repeats until the number of retries is exhausted or the transfer succeeds.

Processes that support ReliableTransfer will have a ReliableTransfer section in the configuration file.

The following parameters are set for the ReliableTransfer option:

Retries	Specifies the number of times to retry transferring data when an error occurs. When this parameter is set to 0, no retries will be attempted and ReliableTransfer is not activated.
TimeBetweenRetries	Specifies the amount of time, in milliseconds, to wait between retries.
OnError	Specifies how to handle the error condition:
Log	Indicates that once all retries have been exhausted and an error still persists, the error will be logged and the process will appear to be successful to the enclosing process flow.
ThrowException	Indicates that once all retries have been exhausted and an error still persists, an exception will be thrown to the enclosing process flow.

Chapter 6

Server Deployment

The topics in this section discuss areas related to deploying and managing servers.

6.1 Connections Tab

The **Connections** tab of the File Systems window is where the servers are displayed and managed.

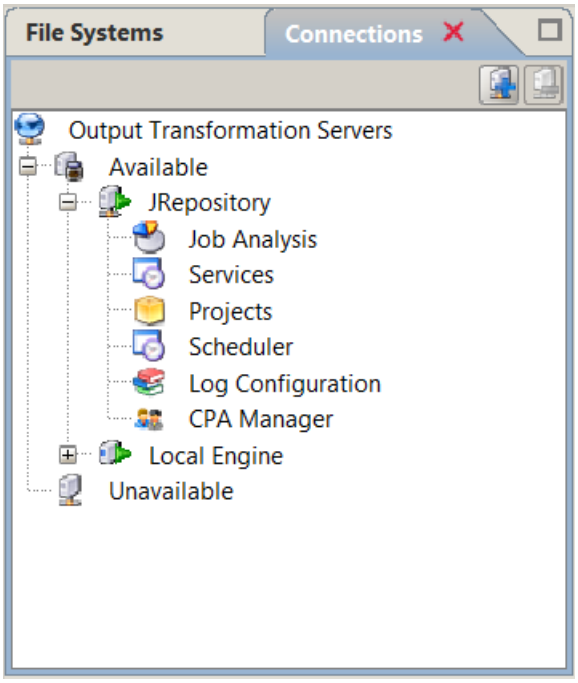


Figure 6-1: Connections tab while connected to an application server

The **Available** tree lists all accessible servers that have been added to Output Transformation Server. By default, the local host engine is always shown in the Available tree and cannot be removed. By right-clicking on your server's root node in the Available tree, a context menu appears with the following options:

- **Connect**. Connects to the server where you have deployed Output Transformation Server.
- **Restart**. Restarts the server.
- **Show Dashboard**. Displays the Dashboard, which relays some basic performance statistics about your application server.

- **Information**. Displays the Information dialog, which contains some basic statistics on your server's uptime and job statistics.
- **Disconnect**. Severs the connection from Output Transformation Server to your application server.

The Connections tab also houses several tools that can be used to monitor various aspects of your application server from within Output Transformation Designer.

Note: If the Connections tab is closed accidentally, you can reopen the tab by navigating to **Window** > **Connections**.

Related Links

- *OpenText Output Transformation Designer - User Guide (VDTOTS-H-UTD)*
- "Dashboard" on page 280
- "Information" on page 281
- "Server Tools" on page 282
- *OpenText Output Transformation Designer - User Guide (VDTOTS-H-UTD)*

6.1.1 Dashboard

The Dashboard is a server tool that gives you many ways to view how well the server is performing while it is running. (For more information on server tools, see "Server Tools" on page 282.) The Dashboard informs you by showing:

- Jobs run today
- Processing time for jobs run today
- Currently running jobs
- Heap sizes for current jobs

6.1.1.1 Dashboard Icons

Icon	Menu Command	Description
	**Export chart data to csv**	Opens the **Export to CSV** dialog, where you select a file location to export chart data into a CSV file.
	**Maximize this chart**	Maximizes the chart to the full size of the screen.
	**Restore chart layout**	Returns chart to the original view.
	**Refresh Dashboard**	Refreshes the page with the latest server statistics and returns to the original view.

6.1.1.2 Showing Jobs Run Today

The **Jobs Run Today** chart displays every deployed job that is running or have run on the current date. Use this chart to know how many jobs are run each day and to determine which jobs fail to run and why.

6.1.1.3 Showing Processing Time

The **Processing Time** chart displays the total processing time for every job shown in the **Jobs Run Today** chart. The time is cumulative and measured in milliseconds. Use this chart to measure how long it takes to run your jobs for the day.

6.1.1.4 Showing Running Jobs

The **Running Jobs** chart displays the total processing time for any jobs that are currently running. This data is useful as a troubleshooting tool to identify jobs that are taking a long time to run.

6.1.1.5 Showing Heap

The **Heap** chart displays the extra memory storage for the current job, in bytes, allocated among the other jobs. This chart is useful for evaluating whether too much memory is being allocated to other jobs.

6.1.2 Information

The **Information** section displays two types of information about Output Transformation Server:

- The **Statistics** page displays statistical information about the server such as:
  - When the server was last started.
  - When the statistics were last reset.
  - How many total jobs were run since the server was last reset.
  - How many total jobs failed since the server was last reset.
  - Average time it takes for each job to run.
- The **License** page displays deployed license information. For more information, please see *OpenText Output Transformation Designer - User Guide (VDTOTS-H-UTD)*.

6.1.3 Server Tools

Various tools are available to handle the administration of all aspects of running a server.

6.1.3.1 Job Analysis

The Job Analysis feature assists you in keeping track of jobs that have been run on the server. It also provides statistics such as start and finish dates, run times, input size, and whether it was successful or not, along with other details associated with the job. There are three different ways that Job Analysis can help you sort through and find jobs:

- Finding Run Jobs Using a Date Range
- Finding Recently Run Jobs
- Finding Run Jobs Using the Query Builder

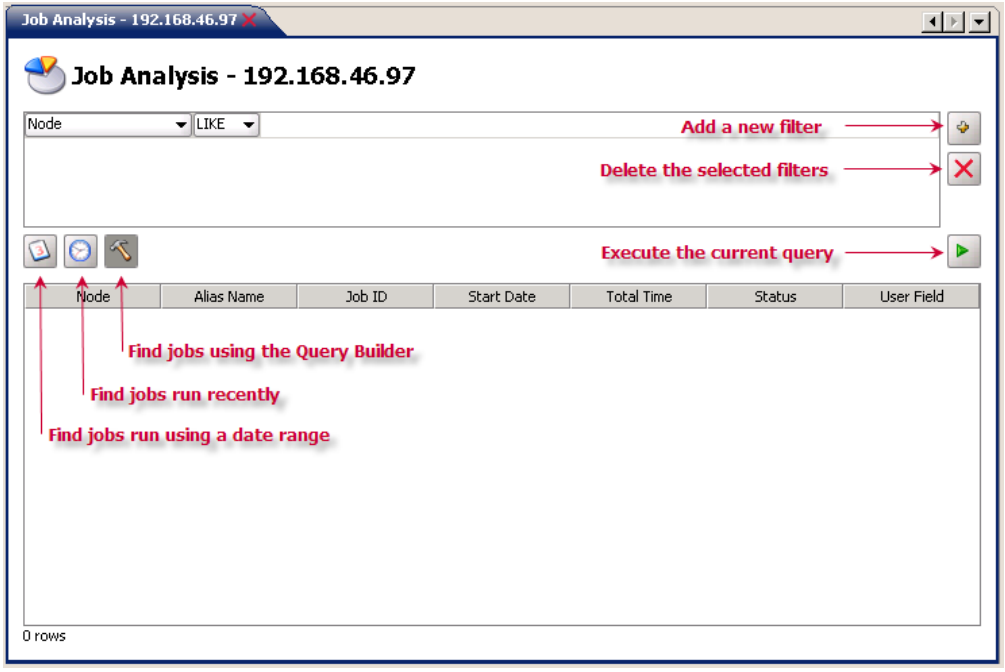


Figure 6-2: Job Analysis window

6.1.3.1.1 Finding Run Jobs Using a Date Range

By inputting two dates, a list of all jobs run between these dates will be generated. To find jobs run using a date range:

1. In the **Job Analysis** view, click the **Find jobs run using a date range** icon to open the **Find jobs run using a date range** dialog.
2. In the **From** field, enter the start date. This date will be the initial point of your search. The date can be entered as "Month Day, Year" format (for example, July 25, 2008), or selected in the calendar dropdown from the calendar icon.
3. In the **To** field, enter the end date. This date will be the end point of your search.
4. Click the **Execute the current query** icon to execute your query, and a list of jobs run matching your criteria will be generated in the table below.

6.1.3.1.2 Finding Recently Run Jobs

A list of jobs run in the selected number of minutes, hours, or days is generated with this element. To find jobs run recently:

1. In the **Job Analysis** view, click the **Find jobs run recently** icon to open the **Find jobs run recently** dialog.
2. In the **Query jobs run in the last** field, enter a number, and define the measurement of time from the dropdown menu to the right of this field. You can select from minutes, hours, or days.
3. If desired, you can further refine your search by using the **Query jobs running longer than** field to find jobs that ran for longer than a defined number of seconds.
4. Click the **Execute the current query** icon to submit your query, and a list of jobs run matching your criteria will be generated in the table below.

6.1.3.1.3 Finding Run Jobs Using the Query Builder

With the Query Builder, you can set up a series of filters to create a customized list of jobs run. To find jobs using the Query Builder:

1. In the **Job Analysis** view, click the **Find jobs using the Query Builder** icon to open the **Find jobs using the Query Builder** dialog.
2. In the first dropdown menu, select the category you want to search within.
3. In the second dropdown menu, select how specific or general the search should be; for a specific search, select **=**, or for a more general search, select **LIKE**.
4. In the text field, enter your search criteria.
5. To further refine your search, you can add or delete filters by selecting the respective icons, and enter more search parameters.

6. Click the **Execute the current query** icon to execute your query, and a list of jobs run matching your criteria will be generated in the table below.

6.1.3.2 Services

The **Services** feature displays a list of all service configuration files currently deployed on the server.

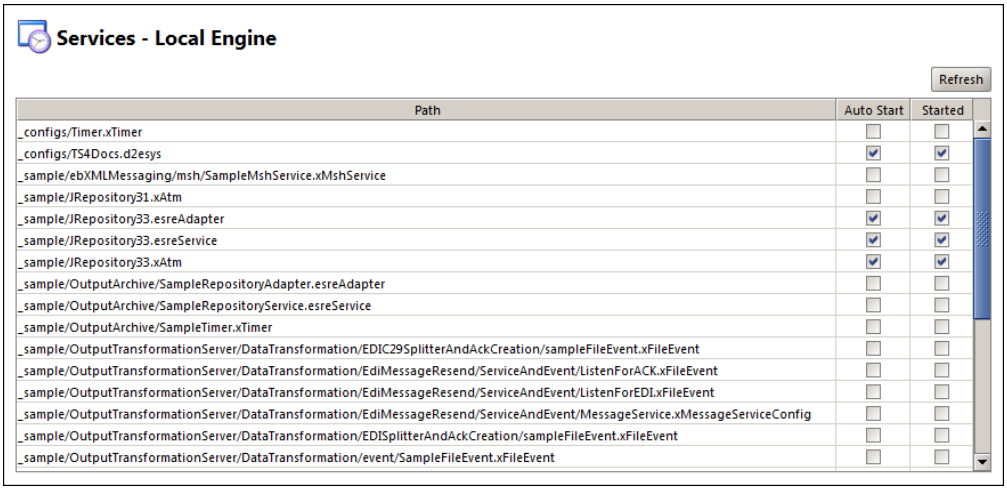


Figure 6-3: Services window

On this screen, you can:

- Click any service's **Auto Start** check box to automatically launch the service the next time the server starts up.
- Click any service's **Started** check box to immediately start the service. Clicking the **Started** check box launches the **Execution Options** dialog, where you can add or remove job variables. After the job variables have been defined, click **Execute** to start the service.

6.1.3.3 Projects

The Projects feature displays a list of all projects currently deployed on the server. You can reorganize the list by clicking the column headings to switch between ascending or descending organizational modes.

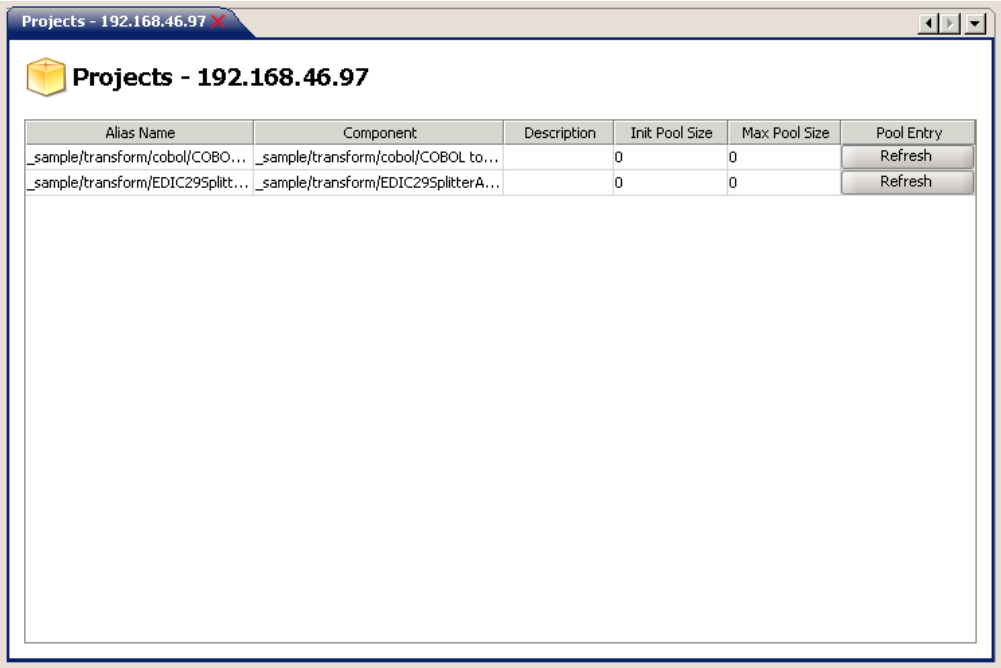


Figure 6-4: Projects window

6.1.3.4 Scheduler

The Scheduler allows users to schedule a component to run daily, weekly, or at certain times. Scheduled components can be made to run only once, or they can be recurring jobs.

6.1.3.4.1 Adding a Scheduled Job

To add a new scheduled job:

In the **Scheduler** view, right-click in the table, and select **Add Schedule** from the context menu.

Note: There are two tabs on this form. Most schedules can be configured on the Basic tab. The Advanced tab is used to set a more precise schedule, including scheduling by day of the month, month, and year (used primarily for month-end or year-end reports). Configure your schedule on one tab only. The last tab with a field value change will be the schedule that takes effect. Do not use both tabs to schedule one job.

6.1.3.4.1.1 Add Schedule Basic Tab

1. Select the type of schedule you want to create using the **Type** dropdown menu. Scheduling a job will run a component from a start time to a stop time, and triggers it at certain intervals. Scheduling a service will run a service component from a start time to a stop time, and uses the service's own built-in triggering mechanism.
2. Specify a unique name in the **Name** field to identify the scheduled job.
3. Select a component to run from the **Config File** drop-down list. The contents of this list will change depending on the **Type** selected.
4. Enter an optional description in the **Description** field.
5. Select or clear the **Enabled** check box. By default it is selected (enabled).
6. Choose the **Days of Week** that this job should run.
7. Select the **Start** and **Stop** times.
8. Enter the **Interval** at which this job should be triggered. This option will only be available when the **Type** selected is **job**.
9. Add any **Job Variables** that the scheduled job may need to run properly. Add and remove Job Variable entries by using the corresponding buttons on the right side of the table view.
10. Test the scheduled job by clicking the **Test** button. If, for some reason, there is an error generating the scheduled job, an error message will be shown with some advice on how to proceed.
11. When the scheduled job tests successfully, click **OK** to finish adding the job.

6.1.3.4.1.2 Add Schedule Advanced Tab

1. Select the type of schedule you want to create using the **Type** dropdown menu. Scheduling a **job** will run a component from a start time to a stop time, and triggers it at certain intervals. Scheduling a **service** will run a service component from a start time to a stop time, and uses the service's own built-in triggering mechanism.
2. Specify a unique name in the **Name** field to identify the scheduled job.
3. Select a component to run from the **Config File** drop-down list. The contents of this list will change depending on the **Type** selected.
4. Enter an optional description in the **Description** field.
5. Select or clear the **Enabled** check box. By default it is selected (enabled).
6. Define the scheduled **start** and **stop** times and dates.

   These fields accept the following wildcards:

   - Seconds: *

- Minutes: *
- Hours: *
- Day of month: *, ?
- Month: *
- Day of week: *, ?
- Year: *, empty

7. Add any **Job Variables** that the scheduled job may need to run properly. Add and remove Job Variable entries by using the corresponding buttons on the right side of the table view.

8. Test the scheduled job by clicking the **Test** button. If, for some reason, there is an error generating the scheduled job, an error message will be shown with some advice on how to proceed.

9. When the scheduled job tests successfully, click **OK** to finish adding the job.

Once the Job or Service has been scheduled, it will appear in the Scheduler, and will also appear in the System Configuration Parameters. It is advised that changes to the schedule be done in the Scheduler, not from the System Configuration Parameters. For more information , see "System Configuration" on page 17

6.1.3.4.1.3 Examples

The Scheduler displays all configured schedules and lists details of each task.

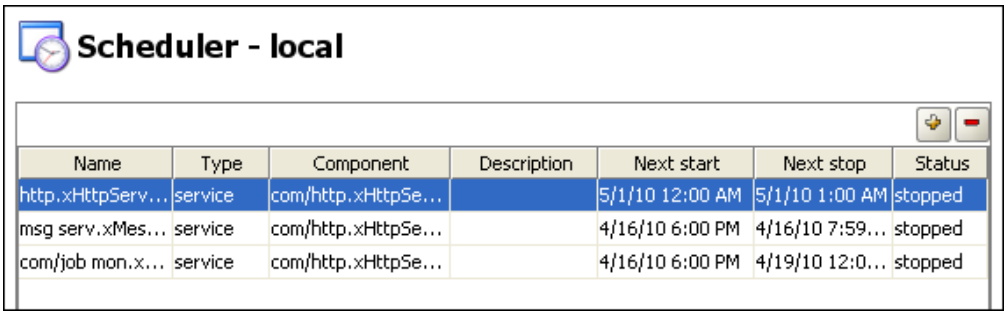


Figure 6-5: Scheduler with 3 scheduled services

Example 6-1: Example 1

The following example is a service scheduled to run weekdays at 6 PM. It is configured on the Basic tab and the results appear in the System Configuration parameters as illustrated.

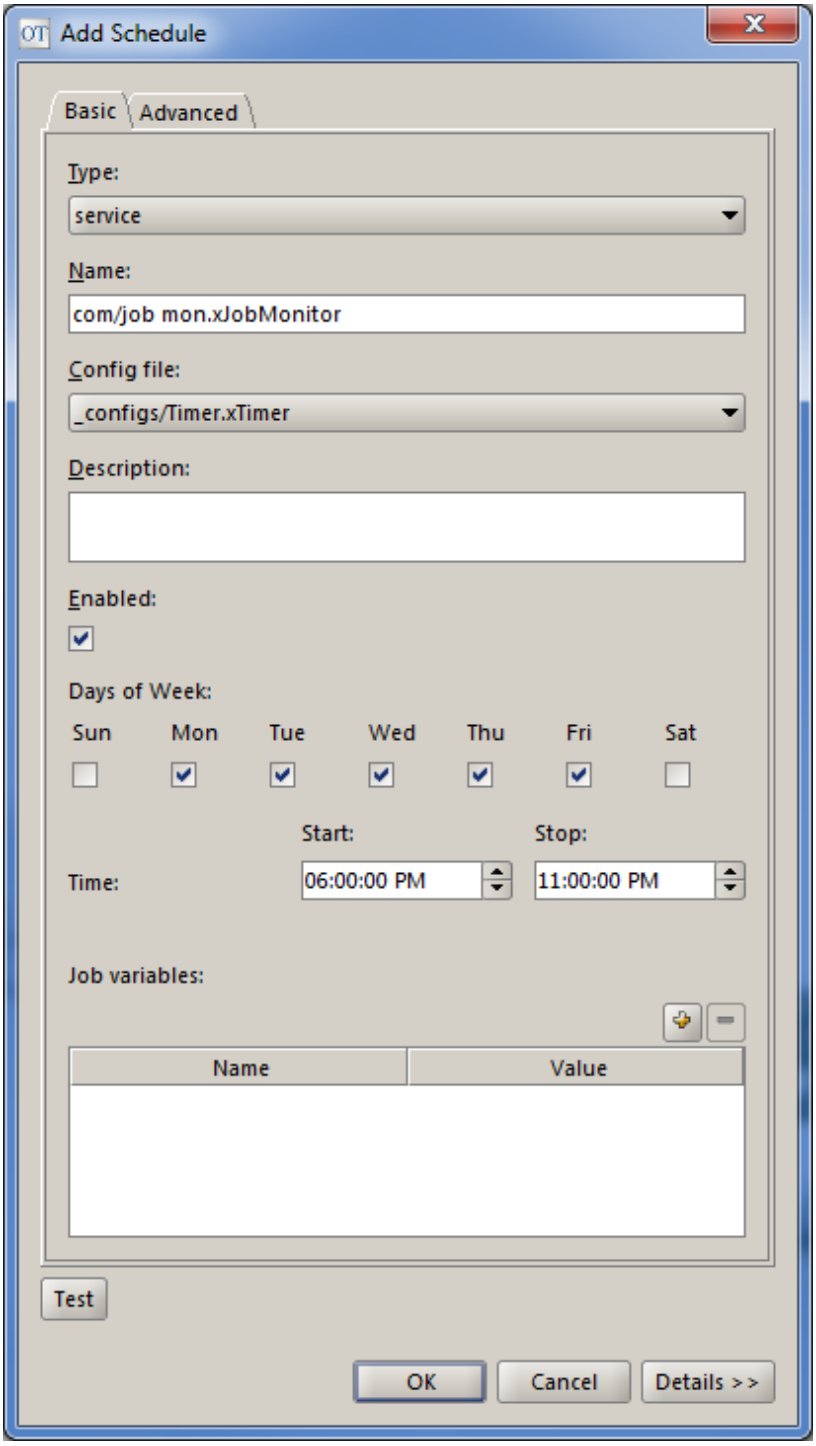


Figure 6-6: Basic tab configuration for a service that runs Monday – Friday at 6 PM (18:00:00)

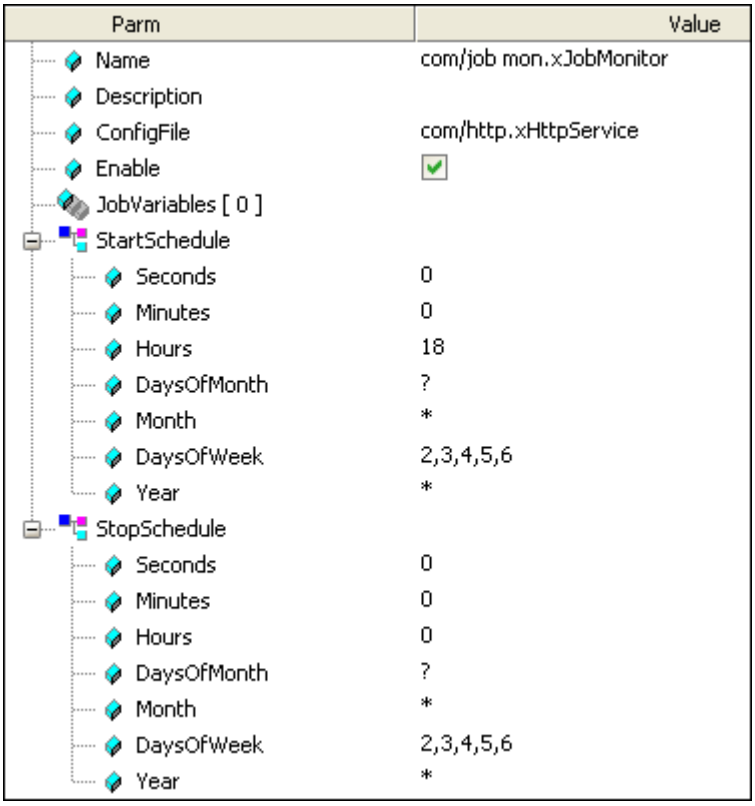


Figure 6-7: System Configuration parameters for a service that runs Monday – Friday at 6 PM (18:00:00)

Example 6-2: Example 2

The following example is a service scheduled to run daily from 8 AM to 10 AM and again from 6 PM to 8 PM. It is configured on the Advanced tab and the results appear in the System Configuration parameters as illustrated.

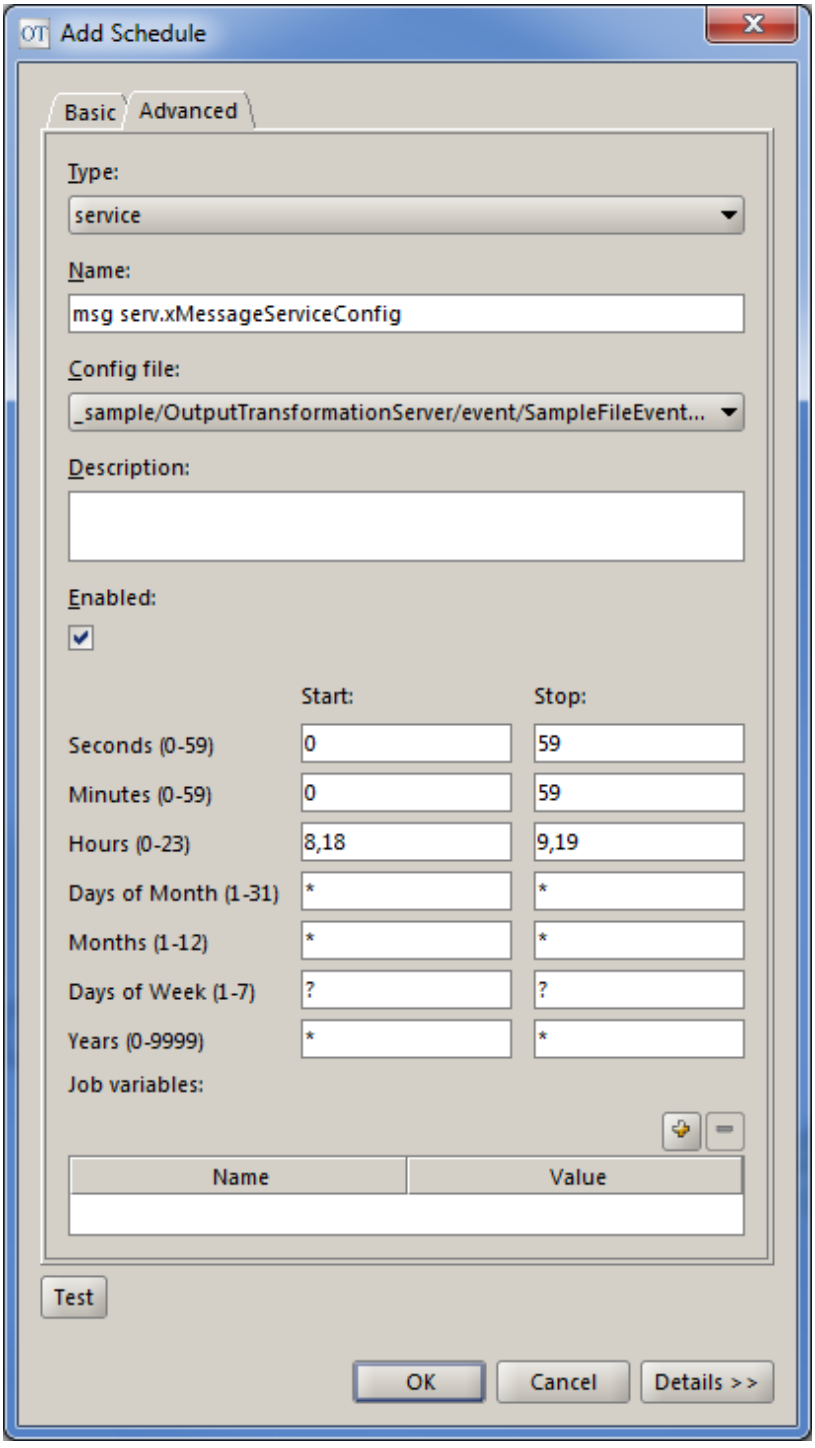


Figure 6-8: Advanced tab configuration for a service that runs daily from 8:00:00 – 9:59:59 and 18:00:00 – 19:59:59

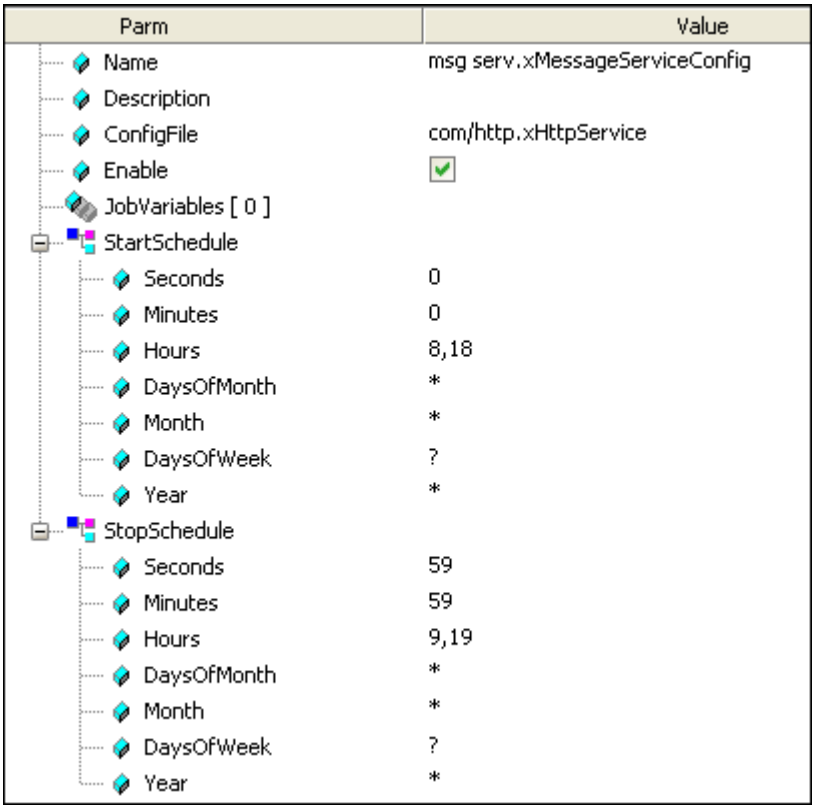


Figure 6-9: System Configuration parameters for a service that runs daily from 8:00:00 – 9:59:59 and 18:00:00 – 19:59:59

6.1.3.4.2 Editing a Scheduled Job

To edit a scheduled job:

1. In the **Scheduler** view, right-click the scheduled job you want to edit, and select **Edit Schedule** from the context menu.
2. Make the necessary changes to the job details, and click **OK** to save your changes.

6.1.3.4.3 Removing a Scheduled Job

To remove a scheduled job:

In the **Scheduler** view, right-click the scheduled job you want to remove, and select **Remove Schedule** from the context menu.

The scheduled job is removed from the table view.

6.1.3.5 Log Configuration

The **Log Configuration** feature allows you to quickly view your Log Profiles at a glance. You can activate and deactivate your various logging profiles from this screen. To view a log, right-click the filename and select **Show Log** from the context menu.

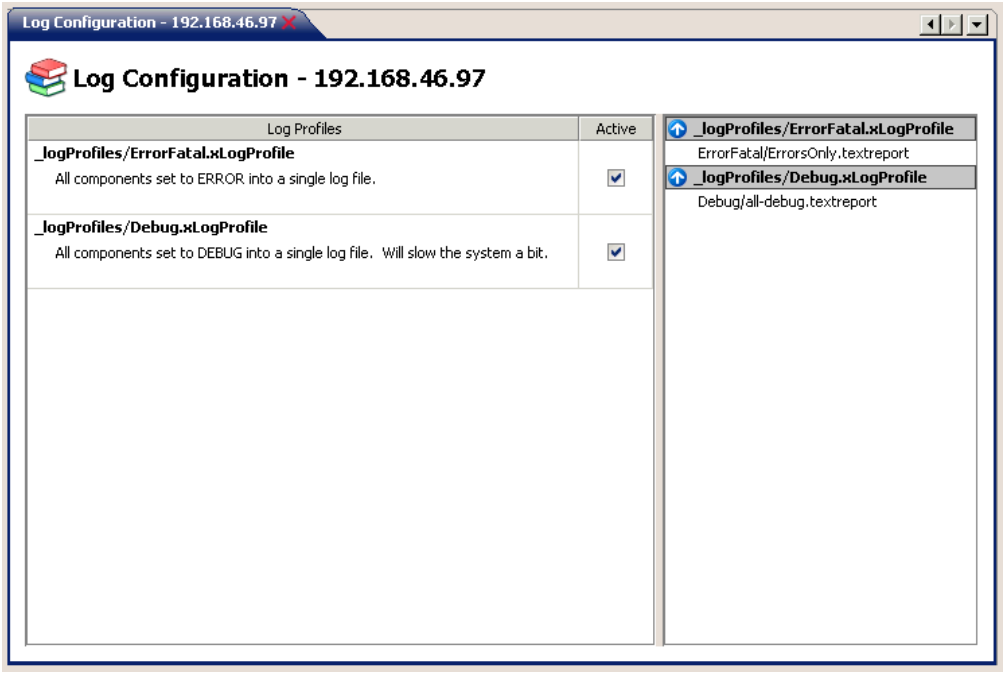


Figure 6-10: Log Configuration window

Related Links

- "Logging Profiles" on page 46

6.2 Setting up a Server

Setting up a server packages Output Transformation Server in an Enterprise Archive (EAR) to run as an enterprise application for deployment to a supported JEE application server. A web application archive (WAR) file can also be created through the Package and Deploy Wizard for any web applications that may require one to facilitate a connection.

Once packaged, the enterprise application contains Output Transformation Server along with your current repertoire of installed OpenText products while the WAR file permits communication between Output Transformation Server and your web application. With the application set up, various server related tasks become available such as the deploying of projects, configuring logs, and performing job analysis.

The packaging process can be easily completed by using either the Package and Deploy Wizard or the Application Server Deployment Set up script.

If you are using WebSphere Liberty, while creating a deployment package you are also provided with an option to automatically deploy the package to your application server.

Note: For Tomcat or JBoss Web Server users with multiple instances, you must ensure that the instances do not share the same shutdown port number because use of the same port number will trigger a bind exception and cause subsequent instances to fail to start. To avoid this, when creating the deployment package if you specify the number of instances you want to create in the wizard or script, the port value number is incremented automatically. If you have already created the deployment package, you can edit the respective `setenv.bat` or `setenv.sh` file and manually change the value of the `ots.shutdown.port` parameter for all instances with the exception of the first instance. JBoss Web Server users can locate this file in their `<install_home>\JBossWebServerBase\<instance_name>\bin\` directory while Tomcat users can locate this file in their `<install_home>\TomcatBase\<instance_name>\bin\` directory.

If you have already created a deployment package and are upgrading to a newer version of the product with the patch method, the deployment package is automatically rebuilt for use with the newer version. For more information, see *OpenText Output Transformation Server - Installation guide (VDTOTS-IGD)*.

6.2.1 Creating a Deployment Package with the Package and Deploy Wizard

To set up a server using the Package and Deploy Wizard:

1. On the **Welcome** screen, click **Package and deploy a new instance**.

   Tip: If the Welcome screen is not initially displayed in the Development window upon starting Output Transformation Designer, selecting **Window** > **Welcome** opens it in a new tab.

2. The **Package and Deploy Wizard** screen provides information on its purpose along with instructions on how to navigate within the wizard. Click **Next**.

3. On the **Select the Target Application Server** screen, you must fill in the information about the server you want to connect to. Depending on the server type, you may not need to complete all the fields. The wizard will not allow you to continue unless all the mandatory fields for your server type are filled in.

Field	Description
Server Type	Indicates the type of application server you wish to connect to.
Server Version	Specifies the version number of the application server.
Server Home	Indicates the installation directory of your selected application server. A default location is provided, but you will have to enter another file path if your application server's files are installed elsewhere.
Instance Name	Designates the name used to identify this particular instance. This name should be distinct as the packaging process will overwrite any existing instances.
Host Name	Identifies the server name or IP address for your application server.
Web Context Root	Specifies the context root location of the Output Transformation Server Manager module. By default, it is set to **OTSManager**. This administration console can be accessed by going to `http://<host>:<port>/<webcontextroot>` in your web browser.
HTTP Port	Declares the HTTP port on the server you wish to connect to.
Is Secure	Specifies whether the application server is set up to use SSL protocols.

Tip: Variables are permitted in the **Host Name** and **HTTP Port** fields so that a single EAR file can be used in clustered nodes.

Once you are finished entering your server information, click **Next**.

4. On the **Provide Server Details** screen, you must supply some server configuration options about your server.

   The **Base Repository** field is the target directory that stores all the configuration files needed for your instance. Output Transformation Server always uses a base repository so if you do not select the **Create Base Repository** check box to promptly establish the repository at the location listed in the **Base Repository** field, then the application will try to create it later if one cannot be found at the specified location. The default base repository location is set to the `<install_home>\BaseRepositories\<appserver>\<instance_name>` directory. While you can change this to a file path of your choosing, your specified `<instance_name>` is always appended to the end of the base repository location. The default location is suitable for most users, but users may opt to reuse a base repository from a previously installed version of the product to take advantage of earlier configuration settings.

   In the **Variables** table, you can enter system-wide variables that are used to store information for retrieval in processes and components. If you have any variables set up in the currently active system configuration, they will appear here. If you wish to add or remove any variables, you can select the  or  buttons, respectively. Variables can also be set to initialize when starting the application server or standalone Output Transformation Server by using the **–D** parameter when calling the JVM.

   In the **Additional Instances** table, you can configure multiple instances of Output Transformation Server for deployment, which will be included in your EAR file or deployment package. If you want to add or remove an instance, you can select the  or  buttons, respectively. When adding an instance, you must also provide a **Name** to identify the particular instance, a **Base Repository** location for the instance to store its configuration files, and the **HTTP Port** the instance uses.

   Once you are finished setting your server details, click **Next**.

5. On the **Configuration Manager** screen, you can set up the Output Transformation Server Configuration Manager, which is a centralized storage system for projects and resources commonly used within an Output Transformation Server cluster and for collaboration between people working on the same projects. Select **Use Configuration Manager** if you want to use Configuration Manager to store common projects and resources.

   If you are using Configuration Manager, you must set the following options that appear:

Field	Description
Enable Automatic Synchronization	Indicates whether your projects and resources are automatically synched with Configuration Manager.
Automatic Synchronization Interval (Min.)	Designates the length of time, in minutes, between synchronizing with Configuration Manager.
Enable Update Cache	Permits the caching of resources, which allows Configuration Manager to run without having to access the database each time it is called upon to check if there is an updated version.
Enable OTS Authentication	Denotes that the authentication engine is enabled.
Administrator User	Indicates the name of the Configuration Manager Administrator account. If you are using the newer authentication scheme with encryption, the account is hard coded to **CMAdministrator**. If you are using the older authentication scheme, the account is hard coded to **Administrator**. (You can verify the user name to use in the `<install_home>\settings\XenosEngine.config` file by reviewing the `registryUser` value.
Current Administrator Password and **New Administrator Password**	Specifies the password to use for the Administrator's account. The first time you configure the server, the password will be blank so you only need to enter a new password in the **New Administrator Password** field. If you need to run the Package and Deploy Wizard again, you will need to change the password. In the **Current Administrator Password** field, enter the password you created during the last setup. Then enter a new password in the **New Administrator Password** field. If you do not change the password, the configuration and deployment will fail.
Database	Specifies the type of database to use for the repository.
Connection	Denotes the type of connection used to connect to your Configuration Manager instance on the application server. You can select from either **JDBC** or **JNDI**.

If you do not want to use the Output Transformation Server Configuration Manager, the base repository directory will be used as the target repository for storing resources instead.

Once you have finished with your selections, click **Next**.

If you are using the Output Transformation Server Configuration Manager, the **Database Setup** screen appears. Proceed to the next step.

If you opted not to use the Output Transformation Server Configuration Manager, the following screen is different depending on your application server type. If you are using Tomcat, JBoss Web Server, or WebSphere Liberty, the **Summary** screen appears and you can proceed to step 10.

6. The information collected on the **Database Setup** screen is used to connect and create the database for the Output Transformation Server Configuration Manager. The following fields are available:

Field	Description
Driver	Denotes the driver for your database type. Click the **Ellipsis** button to display the **Select JDBC Driver** dialog to choose a database from the list.
Database URL	Indicates the server name or IP address of the machine the database is running on. The **Host**, **Port**, and **Database name (SID)** also need to be entered into their respective portions of this field. You can enter them manually into the field or click the **Ellipsis** button to display the **Create JDBC URL**, which assists in the process.
Use Existing Database	Designates that an existing Output Transformation Server Configuration Manager table, created during a previous run, is used. **Note:** This option is not available if you are using an Oracle database.

When you are finished with your configuration, click **Next**.

7. The **Database User** screen assists in establishing a new user or confirming the credentials for an already existing Output Transformation Server Configuration Manager database user.

To create a new user, enter a user name and password into their respective fields.

To use an existing user, select the **Use Existing Database User** check box, and enter the user name and password into their respective fields.

Note: All database users must have read/write access. Additionally, in situations where you are using a pre-existing user but the database itself

does not yet exist, the user must have the proper permissions to create a database and tables within the database server.

The **Test** button checks the database to ensure the login information is correct for an existing user, or that the user name does not already exist in the case of new users.

Once you have entered your database user information, click **Next**.

If you created a new user, the **Database Admin User** screen displays and you can proceed to the next step.

If you used an existing user, the **Select the Packaging Format** screen appears and you can proceed to step 9.

8. On the **Database Admin User** screen, you must enter the database administrator credentials in their respective fields so that the user used/created in the previous screen can be created, if necessary, and granted access to the database.

   Click the **Test** button to verify your user name and password information. If your administrator credentials cannot be authenticated, then you cannot continue with the Package and Deploy Wizard until the correct information is provided.

   When your administrator's user name and password have been verified, click **Next**.

   If you are using Tomcat, JBoss Web Server, or WebSphere Liberty, the **Summary** screen appears and you can proceed to step 10.

9. On the **Select the Packaging Format** screen, you must select the method to package the enterprise application. You can choose to package it as an Enterprise Archive (**EAR**) file or **Unpacked** (a deployable enterprise application directory).

   When you have made your selection, click **Next**.

10. The **Summary** screen outlines your configuration according to the selections you made throughout the wizard and which will form the basis for the enterprise application package. Review your settings and if any changes are required, click **Back** until you reach the appropriate screen containing the settings you want to modify.

    Once you have confirmed your settings, click **Next**.

11. On the **Enterprise Application Packaging** screen, using the information you provided throughout the wizard, the enterprise application is created and packaged. If you elected to create a Output Transformation Server Configuration Manager or package the Reporting Module as a separate WAR file, they are also created during this process. Any web applications you selected to include in your server setup package are also established at this time. Progress messages are displayed in the details area of the dialog and you cannot continue until the wizard is finished processing the files.

Note: The display of details can be enabled or disabled with **Show/Hide Details**.

If the packaging process was successful, the text **BUILD SUCCESSFUL** appears at the end of the progress updates along with the **Total time** it took to generate. If this text does not appear, you can scroll up through the progress details to investigate why the packaging procedure failed before returning to previous screens in the wizard to modify your settings. Moreover, warnings are displayed on the left side of the screen, which may help you isolate the cause of the problem. If the build was successful, click **Next**.

12. Depending on the application server type you selected, different screens appear.

    For JBoss Web Server or Tomcat users, with the enterprise application package created, you are ready to deploy your instance to the server. (For more information about deploying, see "Deploying the Packaged Application to an Application Server" on page 303.) Click **Finish** to close the Package and Deploy Wizard.

    - If you are deploying to a JBoss Web Server machine, the **JBoss Web Server Deployment Steps** screen displays. Your JBoss Web Server deployment package is located in the `<install_home>\JBossWebServerBase\<instanceName>` folder.
    - If you are deploying to a Tomcat server, the **Tomcat Deployment Steps** screen displays. Your Tomcat deployment package is located in the `<install_home>\TomcatBase\<instanceName>` folder.

    For WebSphere Liberty users, with the enterprise application package created, you are ready to deploy your instance to the server. You can choose to automatically deploy your package to your WebSphere Liberty server through the Package and Deploy Wizard by selecting the respective check box for your application server type. (For more information about automatic deployments, see "Performing automatic deployments with the Package and Deploy Wizard" on page 305.

    If you prefer to manually perform the deployment, click **Finish** to close the wizard.

    If you are deploying to a WebSphere Liberty server, the **IBM WebSphere Liberty Deployment Steps** screen displays. Your WebSphere Liberty deployment package is located in the `<install_home>\web\appserver` folder and is named `OTS_<version>-wlp23.0.ear` by default.

6.2.2 Creating a Deployment Package at the Command Line

A script is included with your installation to assist with creating a package for deployment to supported Java EE application servers. It does not provide as many configurable options as the Package and Deploy Wizard, but can gather the necessary deployment parameters to create a basic deployment package for your application server. (For more information on the Package and Deploy Wizard, see "Creating a Deployment Package with the Package and Deploy Wizard" on page 294.)

A script is included with your installation to assist with creating a package for deployment to supported Java EE application servers. You can also choose to allow the script to automatically deploy the package to a WebSphere Liberty application server. It does not provide as many configurable options as the Package and Deploy Wizard, but can gather the necessary deployment parameters to create a basic deployment package for your application server.

The script provides you with the appropriate options to select from and you must type your selections at the command prompt. You can consult the Package and Deploy Wizard topic for more information about the settings you are configuring using the script. Furthermore, some contextual information about your selections and their implications are provided below:

- The setup script is stored in the `<install_home>\maint` folder. Windows users must run the `SetupDeployment.bat` file and Linux users must run the `SetupDeployment.sh` file.
- The script scans your installation directory for all installed versions and you can choose the version you want to set up a deployment package for.
- When making your selections, values shown within the square brackets indicate the default value. If you do not type in a selection for a setting that has a default value assigned to it before pressing Enter, the default is used.
- If you created a JBoss Web Server deployment package, the files for your instance are stored in the `<install_home>\JBossWebServerBase\<instanceName>` folder.
- If you created a Tomcat deployment package, the files for your instance are stored in the `<install_home>\TomcatBase\<instanceName>` folder.
- If you created a WebSphere Liberty deployment package, the EAR file for your instance is stored in the `<install_home>\web\appserver` folder and is named `OTS_<version>-wlp23.0.ear` by default.

Related Links

- "Creating a Deployment Package with the Package and Deploy Wizard" on page 294

6.2.2.1 Creating an Application Server Deployment Package in Silent Mode

For users performing scripted or automated installations, a silent mode deployment operation that bypasses all command prompt inputs and instead, gathers the deployment properties from the `server-config.xml` file is available. The topics in this section describe the steps for involved in setting up silent mode.

Step 1: Pre-populating the Server Configuration File

The deployment properties are collected from the `<install_home>\settings\server-config.xml` file. The file is automatically generated while creating the deployment package so you must run the Package and Deploy Wizard or the application server deployment setup script.

Step 2: Configuring the Start up Properties File

In the `<install_home>\settings` directory, open the `startup.properties` file in your preferred text editor for editing. Within the file, locate the following lines and set their values:

- **ots.deploy.silent=**. Indicates whether silent mode is enabled. To use silent mode, set this value to **true**. By default, it is set to false.
- **ots.deploy.version=**. Specifies the version to use when creating the deployment package. When typing the version number, you must use the same version number as shown in your `<install_home>\install` folder, which follows the `<Version>_<Build_number>_<YYMMDD>` format.

Step 3: Running the Application Server Set up Script in Silent Mode

With the server configuration and startup properties files configured, you are ready to run the set up script. The set up script is stored in the `<install_home>\maint` folder. Windows users must run the `SetupDeployment.bat` file and Linux users must run the `SetupDeployment.sh` file.

For more information on where the created deployment packages are stored, see "Creating a Deployment Package at the Command Line" on page 300.

6.2.3 Preventing unauthorized access to the WSDL

Efforts should be made to limit access to metadata stored by the application to only authorized users. While most of the data is insignificant to external users, some contain sensitive information such as service lists, methods, or versions in use. When this information is not safeguarded, malicious users can use the access to infiltrate and gather data about the environment, which can then be subsequently used to plan and launch hacking attempts.

The following procedure explains how to lock down access to your Axis2 webservices. After applying these settings, users can list services, but will not be allowed to access the WSDL.

Note: You must perform these steps before deploying the WAR file to the server.

To lock down the Axis2 to prevent access to your WSDL:

1. Navigate to the `<OTS_home>\install\<version>\web` directory and open the `axis2.war` archive with your preferred file archiver application.
2. In the archive, navigate to the `\WEB-INF\conf\` directory and open the `axis2.xml` file for editing.
3. Within the `axis2.xml` file, locate the **exposeServiceMetadata** property and edit the value to **false**.
4. **Save** your changes to the file.
5. In the `axis2.war` archive again, navigate to the `axis2-web` directory and either rename or delete the `HappyAxis.jsp` file.
6. Still in the archive's `axis2-web` directory, locate the `index.jsp` file and open it for editing.
7. Within the file, you must either comment out or remove the reference to the HappyAxis.jsp file:

```
<!--
<li><a href="axis2-web/HappyAxis.jsp">Validate</a>
<br/>
Check the system to see whether all the required libraries are in place
and view the system information.
</li>-->
```

8. **Save** your changes to the file.

6.2.4 Reenabling access to the WSDL on Tomcat

Access to the WSDL is restricted for security purposes, however, you can reenable the WSDL to list the services or build clients based on the WSDL.

To reenable the WSDL:

1. Go to the `<OTS_Home>/TomcatBase/<instance>/webapps/axis2/WEB-INF/conf` folder and open the `axis2.xml` file for editing.
2. Within the `axis2.xml` file, you must make the following changes:
   - Set the `disableServiceList` parameter to **false**.
   - Set the `exposeServiceMetadata` parameter to **true**.
3. Save your changes.
4. Restart your Tomcat server.

   Once Tomcat restarts, you can access any of the following pages:
   - http://localhost:8080/axis2/services/listServices
   - http://localhost:8080/axis2/services/JobRunnerService?wsdl
   - Any other WSDL listed

6.2.5 Deploying the Packaged Application to an Application Server

This section examines deploying the Output Transformation Server application onto your application or web server. For the most up-to-date listing of supported application servers, see the Release Notes.

Before continuing with any deployments, it is recommended you review the General Notes and Prerequisites.

General Notes and Prerequisites

In addition to the following notes and prerequisites, also refer to the notes and prerequisites topic specific to your server type.

- Before attempting any deployments, you must have an EAR file (for WebSphere Liberty) or a deployment package (for Tomcat) generated by your installation of Output Transformation Server. The EAR file or packaged application can be created by either running the Package and Deploy Wizard, command line scripts, or the included Apache Ant scripts. For more information about using the Package and Deploy Wizard or the command line scripts, see "Setting up a Server" on page 293.
- When generating your EAR or deployment package, it is important to note your Instance Name and Web Context Root as these will be needed later.

- If you are using a configuration repository, you must ensure that the repository has been configured prior to the creation of the EAR file or deployment package.
- Any references throughout the guide contained between angle brackets (< >) must be substituted with the details as it pertains to your own configuration. For example, if `<install_home>` is shown, you must replace it with the file path location of your Output Transformation Server installation directory; if `<tomcat_home>` is shown, you must replace it with the file path location of your Tomcat installation directory.

6.2.5.1 Performing JBoss Web Server Deployments

6.2.5.1.1 Notes and Prerequisites for JBoss Web Server Deployments

For JBoss Web Server, the Package and Deploy Wizard and packaging scripts create a packaged application instead of an EAR file. In addition to the requirements specified in "General Notes and Prerequisites" on page 303, the following requirement must be met prior to starting the deployment:

- Server with an installation of JBoss Web Server 5.5.

6.2.5.1.2 Deploying the Output Transformation Server Packaged Application to JBoss Web Server

Deployments to JBoss Web Server are automated and are ready to use after running the Package and Deploy Wizard or the command line/shell scripts. Following packaging, you can locate the files for your instance in the `<install_home>\JBossWebServerBase` directory. Within this directory, run the `start-<InstanceName>.bat` file, which sets some required server properties and executes a script to automatically start your JBoss Web Server machine.

After running the start up script, you can verify whether your server has started by trying to access the Output Transformation Server Manager. Open your web browser and type `http://<serverhome>:<port>/<webcontextroot>` into the address bar to access the Output Transformation Server Manager console. (By default, the port number and web context root are set during the Package and Deploy Wizard to `8080` and `ots`, respectively, but you can customize these values during the packaging process.)

6.2.5.2 Performing Tomcat deployments

6.2.5.2.1 Notes and prerequisites for Tomcat deployments

For Tomcat, the Package and Deploy Wizard and packaging scripts create a packaged application instead of an EAR file. In addition to the requirements specified in "General Notes and Prerequisites" on page 303, the following requirements must be met prior to starting the deployment:

- Server with an installation of Apache Tomcat 9.0.x.

6.2.5.2.2 Deploying the Output Transformation Server packaged application to Tomcat

Deployments to Tomcat are automated and are ready to use after running the Package and Deploy Wizard or the command line/shell scripts. Following packaging, you can locate the files for your instance in the `<install_home>\TomcatBase` directory. Within this directory, run the `start-<InstanceName>.bat` file, which sets some required server properties and executes a script to automatically start your Tomcat server.

After running the start up script, you can verify whether your server has started by trying to access the Output Transformation Server Manager. Open your web browser and type `http://<serverhome>:<port>/<webcontextroot>` into the address bar to access the Output Transformation Server Manager console. (By default, the port number and web context root are set during the Package and Deploy Wizard to `8080` and `OTSManager`, respectively, but you can customize these values during the packaging process.)

6.2.5.3 Performing automatic deployments with the Package and Deploy Wizard

When working with WebSphere Liberty, the Package and Deploy Wizard can be configured to automatically deploy the deployment package to the application server. Before you can proceed with any of the procedures in this section, you must go through the Package and Deploy Wizard to create a deployment package. (For more information about using the Package and Deploy Wizard, see "Creating a Deployment Package with the Package and Deploy Wizard" on page 294. For more information about creating a deployment package at the command prompt, see "Creating a Deployment Package at the Command Line" on page 300.)

Note: The `SetupDeployment.bat/.sh` script also offers an option to automatically deploy the package to the server. To deploy the package, you must provide similar information about the application server as the Package and Deploy Wizard method below.

Performing automated deployments for WebSphere Liberty

To automatically deploy the package to a WebSphere Liberty server from the Package and Deploy Wizard:

1. After you have successfully created a deployment package, on the **IBM WebSphere Liberty Deployment Steps** screen select the **Run WebSphere**

Liberty Automatic Deployment check box and the following boxes appear that you must complete about your application server:

- **Server Name**. Indicates the name of the WebSphere Liberty server where the application will be deployed.

2. When the deployment is complete, you can close the Package and Deploy Wizard by clicking **Finish**.

6.2.5.4 Performing WebSphere Liberty deployments

6.2.5.4.1 Notes and prerequisites for WebSphere Liberty deployments

In addition to the requirements specified in "General Notes and Prerequisites" on page 303, the following notes must be reviewed and requirements must be met prior to starting the deployment:

- Server with an installation of WebSphere Liberty that supports the Java EE 8.0 Full Platform feature

6.2.5.4.2 Deploying the packaged application to a WebSphere Liberty server

This section details deploying the packaged application to a WebSphere Liberty server with autodeploy or manually.

Note: Before starting, make sure you have WebSphere Liberty installed on your machine.

1. At the command prompt, go to the `<wsliberty_home>\wlp\bin` folder.
2. In this folder, run the command to create a server. At the command prompt, type `server create <server_name>` where *<server_name>* is the name you want to assign to the server.

   Note: If you do not specify a server name, then `defaultServer` is used.

   The server is created and the server configuration files are added to the `<wsliberty_home>\wlp\usr\servers\<server_name>` folder.

3. In the `<wsliberty_home>\wlp\usr\servers\<server_name>` folder, open the `server.xml` file in a file editor and verify that the Java EE 8.0 feature has been declared inside the `featureManager` element:

   ```
   <featureManager>
        <feature>javaee-8.0</feature>
   </featureManager>
   ```

   If the feature has not been declared in the file, add it to the file.

4. You are ready to create the deployment package. If you want to automatically deploy the package to the server, you can either run the Package and Deploy Wizard or the `SetupDeployment.bat/.sh` script at the command line. Make sure you enable the automatic deployment to the WebSphere Liberty server option.

For more information about creating the deployment package, see "Creating a Deployment Package with the Package and Deploy Wizard" on page 294 and "Creating a Deployment Package at the Command Line" on page 300.

In place of automatically deploying the package to the WebSphere Liberty server, you can also manually set the server configuration. If you are using the manual deployment method, proceed to the next step.

If you used the automatic deployment method, skip ahead to step 7.

5. For a manual deployment, you must go to the `<wsliberty_home>\wlp\usr\servers\<server_name>` folder and open the `server.xml` file in a file editor. In the file, add the following XML elements while replacing the variables with values as it pertains to your environment:

```
<library id="OTS_LIB_COMMON">
<fileset dir="<ots_home>/<instance_name>/install/<version>/lib/common"
excludes="jakarta.activation*.jar, jakarta.annotation*.jar, jakarta.inject*.jar,
jakarta.json*.jar, jakarta.persistence*.jar, jakarta.servlet*.jar,
jakarta.ws*.jar, jakarta.xml.*.jar*,*jaxb-*.jar, jaxws*.jar, javax.mail.jar,
jsr*.jar, Java-WebSocket*.jar, *osgi*.jar, stax*.jar" includes="*.jar"/>
<fileset dir="<ots_home>/custom/jars" includes="*.jar"/>
</library>
<application id="OTSServer" location="<ots_home>/web/appserver/<version>-
wlp22.0.ear" name="OTSServer" type="ear">
<classloader delegation="parentLast" privateLibraryRef="OTS_LIB_COMMON"/>
<web-ext moduleName="axis2.war"/>
<web-ext moduleName="Xenos-es-admin.war"/>
<web-ext moduleName="Xenos-es-rest-web-app.war"/>
<web-ext moduleName="Xenos-accessibility-atm.war"/>
<web-ext moduleName="Xenos-dashboard.war"/>
<web-ext moduleName="Xenos-remediation-console.war"/>
<web-ext moduleName="Xenos-remediation-rest.war"/></application>
```

Note: You can see an example of this file in the `<ots_home>\web\appserver` folder.

6. Next, you must verify whether the `jvm.options` file exists in the `<wsliberty_home>\wlp\usr\servers\<server_name>` folder. If the file does not exist, you must create it and add the following JVM properties:

```
-Xmx2048m
-Dots.install=<ots_home>
-Dots.startup.mode=appserver
-Dots.home=<ots_base>
-Dots.instance=<instance_name>
```

You must set the variable names for your environment where

<ots_home> is the folder containing the version of Output Transformation Server to use for this instance. For example, a valid value is the `C:\OTS\OTS_BASE\install\22.4.DEV.00_0000` folder.

<ots_base> is the base installation folder for your Output Transformation Server instance on the server. For example, a valid value is the `C:\OTS\OTS_Base` folder.

<instance_name> is the name of the instance to connect to.

Note: You can see an example of this file in the `<ots_home>\web\appserver` folder.

7. Once the package has been automatically or manually deployed, you can start your WebSphere Liberty server. At a command prompt, run the following command:

   ```
   <wsliberty_home>\wlp\bin\server start <server_name>
   ```

6.2.6 Undeploying a Deployment Package from an Application Server

Undeploying a deployment package from the various application server types is discussed in this section.

6.2.6.1 Undeploying the Packaged Application from a JBoss Web Server

As the Output Transformation Server properties required by JBoss Web Server during start up are referenced from within your installation directory, no deployment files are copied to your JBoss Web Server installation folder. Hence, there is no undeployment process so you can simply shut down the server.

6.2.6.2 Undeploying the Packaged Application from a Tomcat Server

As the Output Transformation Server properties required by Tomcat during start up are referenced from within your `<install_home>` location, no deployment files are copied to your Tomcat installation folder. Hence, there is no undeployment process so you can simply shut down the server.

6.2.6.3 Undeploying the Packaged Application from a WebSphere Liberty Server

To undeploy the application EAR with the WebSphere Integrated Solutions Console:

1. On the Welcome screen of the Integrated Solutions Console, navigate to **Application** > **Application Types** > **WebSphere enterprise applications**.
2. On the Enterprise Applications screen, select your EAR file's respective check box.
3. Click **Uninstall**.
4. Optionally, you can also remove the JVM arguments added during the creation of the shared library.

6.3 Connecting and disconnecting application servers

6.3.1 Connecting to a server

After packaging your enterprise application and deploying it to your application server, you can add the server to Output Transformation Server for remote administration. (For more information on packaging and deploying your application, see "Setting up a Server" on page 293.

To connect an application server to Output Transformation Server:

1. On the **Connections** tab, either:

   - Right-click the **Output Transformation Servers** root node and from the context menu that appears, select **Connect**.
   - Click the **Connect to Output Transformation Server**,  , button.

   The **Connect** dialog displays.

2. On the **Host** tab, you must enter the following information:

   - **Alias.** Indicates the name used to identify the server. This field is mandatory.
   - **Host Name.** Designates the address of the server you are adding.
   - **Port.** Designates the port number for the server you are adding.
   - **User.** Specifies the user name required to gain access to the server. The default value is **Administrator**.
   - **Password.** Specifies the password associated with the user account.

3. Optionally, there are some more options that can be set on the **Advanced** tab. For most users, the default settings are appropriate and do not need to be modified. The following configuration settings are available:

   - **Context Root.** Appoints the web context root for the OpenText Output Transformation Server Manager and the default value is **csadmin**. For example, in the URL for the Output Transformation Server Manager, `http://localhost:8080/csadmin`, the context root is `csadmin`.
   - **Auto Connect.** Stipulates whether to automatically connect to this server when Output Transformation Designer is started. By default, this **true**.
   - **Is Secure.** Specifies whether the application server is set up to use SSL protocols.

4. If you want to check whether your server connection is functional and your user credentials are correct, click **Test**. This step is optional so if you do not want to test your connection, you can skip to the next step.

Note: You may need to click the **Details** button to see the information and messages from the connection test.

5. Click **OK**.

   If your settings are correct, the server is added to the **Available** tree on the **Connections** tab and a green arrow appears beside the server name. This means that the server is ready to be used, and you can start deploying projects (project files, events, and process flows) to the server using Output Transformation Designer.

   If your connection was unsuccessful, review the **Details** area of the Connect dialog for error messages that can help you determine the cause of the connection failure.

6.3.2 Disconnecting from a server

To disconnect from a server through Output Transformation Server, on the **Connections** tab either:

- Right-click your server's root node under the Available tree and from the context menu that appears, select **Disconnect**.
- Select your server's root node and click the **Disconnect from Output Transformation Server**,  , button.

The connection to your server is closed and the server is removed from the list of Available servers.

6.4 Deploying a project or resource to the server

Output Transformation Server runs projects, processes and process flows created from within Output Transformation Designer.

There are two methods of deploying a project or resource to a server:

- Output Transformation Designer direct deployment to the server. For more information, see "Deploying a project or resource to the server through Output Transformation Designer" on page 311.
- Output Transformation Designer deployment to the OpenText Output Transformation Server Configuration Manager, if installed and set up. For more information, see "OpenText Output Transformation Server Configuration Manager" on page 325.

Note: If you are modifying your Axis2 configuration to prevent access to the WSDL, you must set up the configuration prior to deployment of the package to the server.

6.4.1 Deploying a project or resource to the server through Output Transformation Designer

Prior to deploying any projects or other resources, like Events or Services, to the server, you must first deploy your application server.

To deploy a project or resource to the application server through Output Transformation Designer:

1. On the **File Systems** tab, navigate through the directories to locate the project or resource file you want to deploy.
2. Right-click on the project or resource file and in the context menu that appears, select **Tools** > **Deploy to Output Transformation Server**.

   The **Deploy to Output Transformation Server** dialog displays.
3. On the Deploy to Output Transformation Server dialog, the **Deployment** table shows what **Resources** are currently available for deployment. Select the **Deploy** check box for all the projects and resources you want to deploy to the application server. Event and Service resources have additional options that can be selected; you can choose to have them **Start** immediately after deployment to the server or opt for them to **Auto** start whenever your application server is started.

If you want to allow your resources to automatically reload with the most recent version whenever an older copy exists in the component pool, select the **Refresh Component Pool** check box.

If you have multiple servers deployed, you must select which server you want to deploy to from the listing in the **Connections** table. To help you distinguish between servers, the **Alias** and **Host** address for the connected servers are displayed in this table.

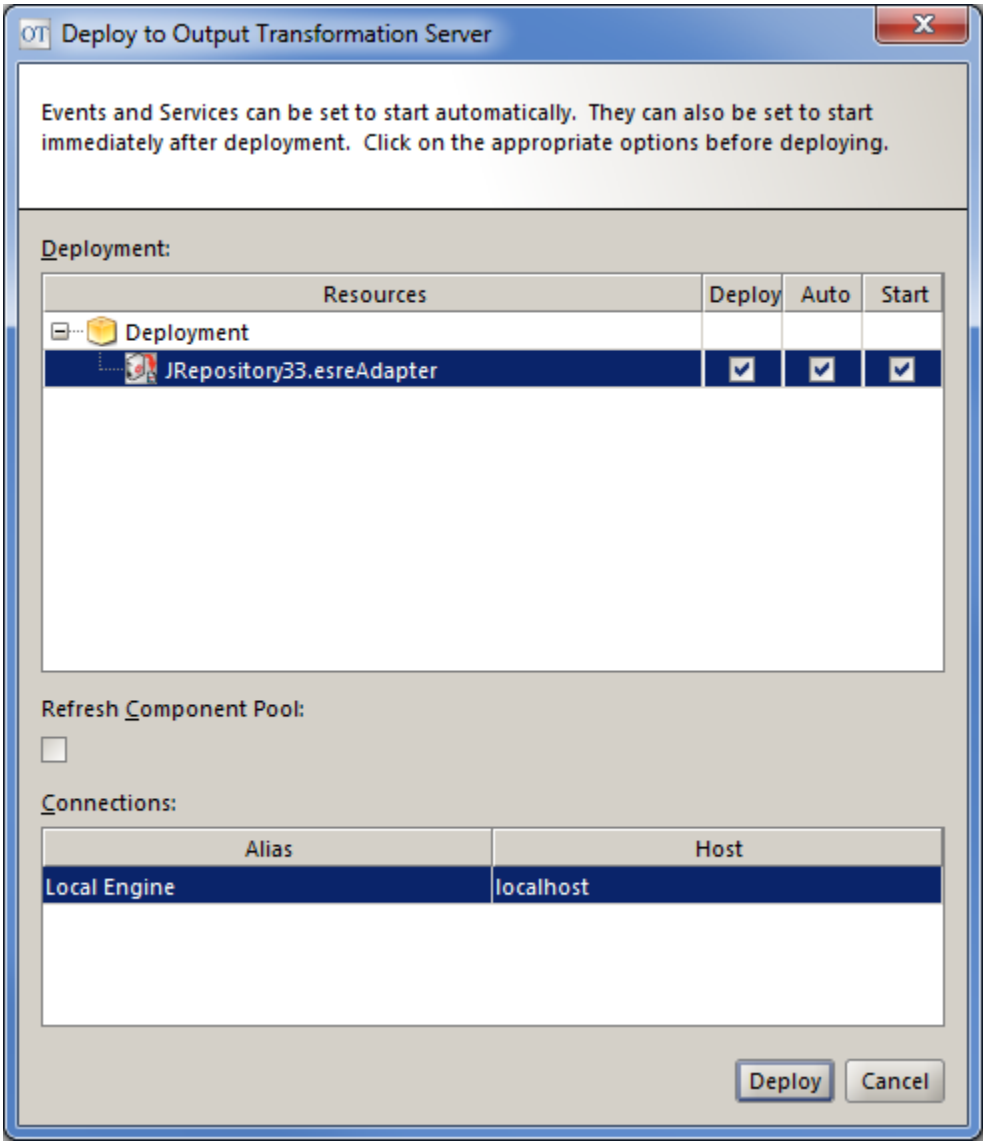


Figure 6-11: Deploy to Output Transformation Server screen

When you are finished making your selections, click **Deploy**.

The **Deploying Projects** dialog appears and displays the deployment's progress. A message will appear indicating whether or not your deployment was successful. If your deployment failed, review the error messages and correct the issue before attempting the deployment again. Furthermore, some extra information about your deployment can be obtained by clicking the **Details** button.

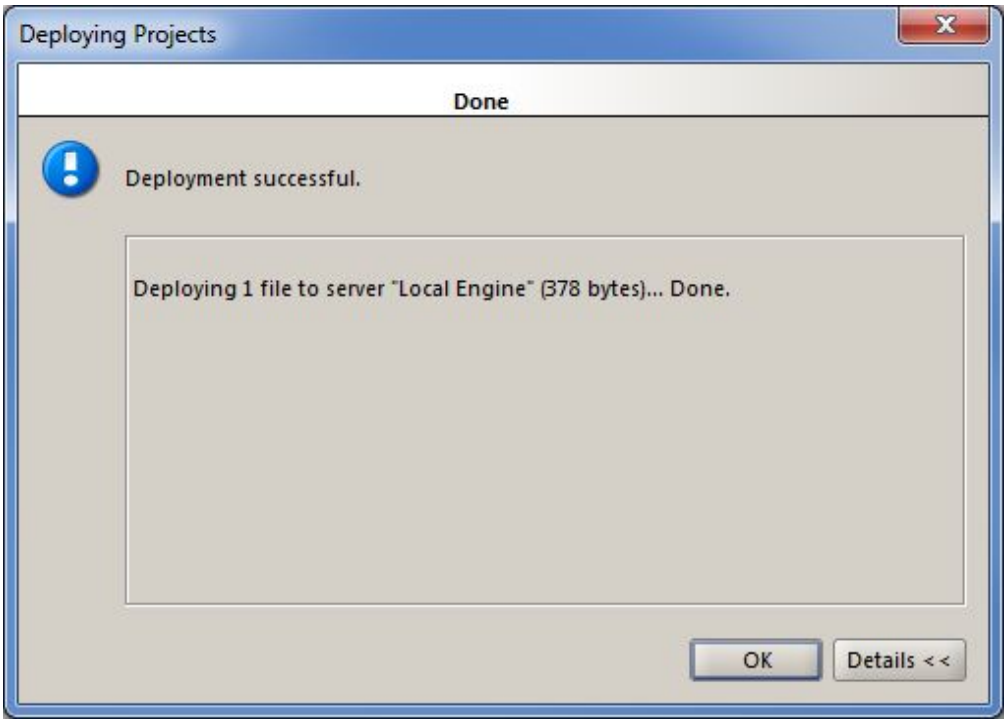


Figure 6-12: Deploying Projects dialog box showing the successful deployment message and expanded Details

Tip: You can verify what resources have been deployed to your server by using the **Services** and **Projects** server tools, available on the Connections tab for deployed servers.

Related Links

- "Events" on page 158
- "Services" on page 250
- "Setting up a Server" on page 293
- "Connecting and disconnecting application servers" on page 309
- "Setting Java Variables for an Application Server" on page 317
- "Running the Deployed Project" on page 318
- "Server Tools" on page 282

6.4.2 Deploying an instance to a remote server

Many production environments maintain separate development workstations from their production workstations, which are hosted on remote application servers. The following procedure illustrates how this works when deploying to a remote Tomcat instance that is running on a Linux server, but the principle behind this procedure can be applied for deployments to remote instances for other supported application servers. There are a few separate parts involved in setting up your servers and they are discussed in the subsequent sections.

6.4.2.1 Prerequisites

Prior to starting these configuration steps, it is assumed that you already have met the following requirements:

- Installation files for Output Transformation Server. By the end of this procedure, the application will be installed on both the development machine and the remote application server.
- Installation files for a supported application server. In this example, Tomcat is used.
- For the purposes of this procedure, the Output Transformation Server installation directory on Windows is under the `C:\OpenText\OTS` directory and will be referenced as `<OTS_HOME_WIN>` while the installation directory on Linux is under `/opt/opentext/ots` and will be referenced as `<OTS_HOME_LNX>`.

6.4.2.2 Step 1: Installing the Tomcat application server on the development machine

You must install the application server on your development machine so that some required property and configuration files can be obtained. Once you have finished with the installation and retrieved the files, you can delete the installation.

1. On your development machine, create a new directory for where you will eventually install Output Transformation Server. You can name this directory anything you want, but for this example, `C:\OpenText` is used.
2. Copy your compressed Tomcat installation file to this directory. (Typically, the name of the installation files follow the `apache-tomcat-<tomcat_version>-windows-<windows_version>.zip` format.
3. Extract the contents of the compressed file in this directory.
4. Rename the extracted directory to `tomcat9`.

6.4.2.3 Step 2: Installing Output Transformation Server on the development machine

1. Copy the base installation file for Output Transformation Server, `OTS_Base.zip`, to `<OTS_HOME_WIN>` and extract its contents to this location.
2. Under your `<OTS_HOME_WIN>` directory, create a new directory called **modules**.
3. Copy your product installation module files to the `modules` folder.
4. Open a command window, navigate to the `maint` folder and run the `OneTimeSetup.bat` file, which expands the compressed build files in the `modules` folder.
5. Optionally, copy any custom jars you are using to the `custom\jars` directory.

6.4.2.4 Step 3: Creating a deployment package on the development machine

In Output Transformation Designer, run through the **Package and Deploy Wizard** to create an enterprise application package for deployment to your chosen application server. (See "Setting up a Server" on page 293 for more information on using the Package and Deploy Wizard.)

While working through the Package and Deploy Wizard, you must fill in the properties according to your setup but ensure these following settings are used for these fields:

- On the **Select the Target Application Server** screen:
  - **Web Context Root**. Accept the default setting of **OTSManager**.
  - **Server Home**. Denotes the application server installation directory that was created in a previous step. In this example, `C:\OpenText\tomcat9` is used.
  - **Is Secure**. Leave unchecked.
- On the **Provide Server Details** screen:
  - **Base Repository**. Replace the default value with the updated `/opt/opentext/ots/BaseRepositories/tomcat/ots` value.
  - **Create Base Repository**. Leave unchecked.
- On the **Configuration Manager** screen:
  - **Use Configuration Manager**. Leave unchecked.

6.4.2.5 Step 4: Installing the Tomcat application server on the production machine

You must install the application server on your production machine so that some required property and configuration files can be obtained.

1. On your production machine, create a new directory for where you will eventually install Output Transformation Server. You can name this directory anything you want, but for this example, `/opt/opentext` is used.
2. Copy your compressed Tomcat installation file to this directory. (Typically, the name of the installation files follow the `apache-tomcat-<tomcat_version>.tar.gz` format.
3. Extract the contents of the compressed file in this directory.
4. Rename the extracted directory to `tomcat9`.

6.4.2.6 Step 5: Installing Output Transformation Server on the production machine

1. Copy the base installation file for Output Transformation Server, `OTS_Base.zip`, to `<OTS_HOME_LNX>` and extract its contents to this location.
2. Under your `<OTS_HOME_LNX>` directory, create a new directory called **modules**.
3. Copy your product installation module files to the `modules` folder.
4. Open a command window, navigate to the `maint` folder and run the `OneTimeSetup.sh` file, which expands the compressed build files in the `modules` folder.
5. Optionally, copy any custom jars you are using to the `custom\jars` directory.

6.4.2.7 Step 6: Configuring the connection between development and production machines

Next, to facilitate the connection you must copy some property files between your development and production machines.

1. From your development machine (`<OTS_HOME_WIN>`, in our example), copy the following directories and files to your production machine (`<OTS_HOME_LNX>`).
   - `BaseRepositiries/tomcat/ots`
   - `TomcatBase/`
   - `web/`
   - `settings/baseRepositoryLocations.properties`
   - `settings/tomcat_ots.properties`

2. In the `settings/baseRepositoryLocations.properties` file that was copied, verify the `baserepo.appserver.ots` value and ensure it is correct for your environment.
3. On your production machine, run **dos2unix** on the `TomcatBase/start-ots.sh` file and validate your `CATALINA` values.
4. Next, run **dos2unix** on the `TomcatBase/ots/bin/setenv.sh` file and validate the `CATALINA` and `OTS` values.
5. If you are running Linux in Headless mode, for the CATALINA_OPTS value add:

   ```
   -DDjava.awt.headless=true
   ```

6.5 Setting Java Variables for an Application Server

When creating an `.ear` file within Output Transformation Designer, you can reference Java system properties as variables for the setup of the host name, HTTP port and JNDI port.

For example, if you start your application server with the following settings:

- `Dxenos.hostname=localhost`
- `Dxenos.http.port=8080`

and you set the host name to `${xenos.hostname}` and HTTP port to `${xenos.http.port}`, Output Transformation Server will use `localhost` and `8080` as the values for host name and HTTP port. That way, when you set up multiple nodes within the cluster you don't need to create an `.ear` file for each server. This means the values for these system variables can be passed at startup, which makes deployment across multiple nodes easier to manage and simpler to do.

For more information on deployment or packaging an `.ear`, see "Deploying the Packaged Application to an Application Server" on page 303.

To use Java system properties defined in an application server:

1. Start the **Package and Deploy Wizard**.
2. Click **Next**.

   The **Select the Target Application Server** page opens with the current values for each variable.

   Note: To define numeric values for the HTTP port or JNDI port, simply enter the port's numeric value in the related text box.

3. To define Java variables for the host name, HTTP port, and/or JNDI port, click the **Ellipsis** button, …, beside their related text boxes.

   The **Variable Browser** dialog box for the variable you chose opens with the variables that have been set in the currently running JVM.

4. To define the variable, you have the option to either:

   - Double-click the predefined variable name or its value in the existing **Variables** list to add the variable name to the **Variable** text box.
   - Enter a new Java variable name directly in the **Variable** text box.

   Note: Type the variable name without using $ or {}, as the Package and Deploy Wizard automatically applies these characters to the name when you save it. For example, if you enter `xenos.http.port`, the Package and Deploy Wizard will automatically store `${xenos.http.port}` as the variable name.

5. Click **OK** to save your changes.

   The new variable name is displayed in the text box for the variable you chose.

6. Click **Next** to continue with your server setup.

The target application server setup creates a `server.info` Java class, similar to the example below, which includes the Java variables:

```
<ServerInfo po-class="com.xenos.j2ee.server.parm.ServerInfoParm"
   serverType="WebSphere"
   version="4.2.2">
<jndiProperties name="java.naming.provider.url"
   value="jnp://${xenos.webservice.hostname}:${xenos.webservice.jndiport}"/>
<jndiProperties name="java.naming.factory.initial"
   value="org.websphere.security.jndi.LoginInitialContextFactory"/>
<ejbProperties name="ejb.lookup.name"
   value="XenosSystemExecute"/>
<jmsProperties name="jms.connection.factory"
   value="ConnectionFactory"/>
<jmsProperties name="jms.source.queue"
   value="queue/SourceQueue"/>
<jmsProperties name="jms.result.queue"
   value="queue/ResultQueue"/>
<webServiceProperties name="ws.wsdl.url"
   value="http://${xenos.webservice.hostname}:${xenos.webservice.port}/ws/services"/>
```

Related Links

- "Setting Up Server Clustering and Load Balancing" on page 40
- "Setting up a Server" on page 293

6.6 Running the Deployed Project

When running a deployed project via the **SystemExecute Session Bean**, **WebService**, or directly via the **Output Transformation Server API**, you must supply the following:

- **Name of the runnable component**. This is the path to the project file, process or process flow.
- **One or more input streams**. If the project requires more than one input, you must use a HashMap to name the streams. Otherwise, you can specify a string or a byte[].

- **One or more output types**. You have the choice of having a string or byte[] returned. If the project creates multiple output streams, these have to be named in a HashMap.
- Optional **Map of name/value pairs** treated as job variables when the project runs.

For an example of how to run the installation verification test, RunJob.java, using a SessionBean, JMS Queue, WebService or directly with the OpenText Output Transformation Server API (no application server required) see the sample source code located under the `\sampleApplication\src\java\com\xenos\tester\RunJob.java` path.

6.7 Setting up Tomcat as a Windows Service

Users in a production environment may want to set up Tomcat as a Windows service. There are a few steps involved in setting up Tomcat as a Windows service for an instance of Output Transformation Server, with an option to perform the installation and configuration of the service either manually or automatically through a script.

6.7.1 Prerequisites

Prior to starting these configuration steps, it is assumed that you already have met the following requirements:

- Installation of Apache Tomcat 9.0.x. For the purposes of this configuration, references to this directory will be referred to as <*Catalina_home*>. A few different versions are available so you must ensure you download a version of Tomcat containing the Windows service wrapper.
- Installation of Output Transformation Server. For the purposes of this configuration, references to this directory will be referred to as <*install_home*>.

Once these prerequisites are met, you can proceed to "Step 1: Packaging an Output Transformation Server Instance for Deployment to Tomcat" on page 319.

6.7.2 Step 1: Packaging an Output Transformation Server Instance for Deployment to Tomcat

1. In Output Transformation Designer, run the **Package and Deploy Wizard** to create an enterprise application package for deployment to your Tomcat server. (For more information on using the Package and Deploy Wizard, see "Setting up a Server" on page 293.)

   The files for your instance are stored in the `<install_home>/TomcatBase/<instanceName>` directory.

2. If you require multiple instances, be sure to specify them in the Package and Deploy Wizard with different instance names for each. Each instance you create will be added to the `<install_home>\TomcatBase` folder.

3. There are two methods of installing and configuring the Windows service; you can opt for either manual configuration or via a script. Proceed to the respective section for the next step:

   - "Step 2A: Installing and Configuring the Tomcat Service (Manually)" on page 320
   - "Step 2B: Installing and Configuring the Tomcat Service (via Script)" on page 321

6.7.3 Step 2A: Installing and Configuring the Tomcat Service (Manually)

Note: If you prefer, the installation and configuration of the Tomcat service can also be completed by running a script instead of manually arranging the settings. For more information, see "Step 2B: Installing and Configuring the Tomcat Service (via Script)" on page 321.

For each instance you created, you must establish a separate Tomcat service.

1. The Tomcat Windows service wrapper is used to create the new service for your instance. To run it, at a command line prompt navigate to `<Catalina_home>\bin` and type:

   ```
   service.bat install Tomcat9<instanceName>
   ```

2. Once the new Tomcat service is created, you are ready to configure the service. At the command line prompt, type:

   ```
   tomcat9w.exe //ES/Tomcat9<instanceName>
   ```

   An **Apache Tomcat x.x Properties** dialog for your service opens.

3. Some configuration parameters about your Output Transformation Server environment must be entered for the Tomcat service properties. You can obtain these values from the `<install_home>\TomcatBase\<instanceName>\bin\setenv.bat` file by opening the file using a code editor. The following parameter values shown in this file are needed:

   - `OTS_HOME=`
   - `OTS_VERSION=`
   - `OTS_INSTALL=`
   - `ots.port=`
   - `ots.shutdown.port=`
   - `ots.host=`
   - `ots.instance=`

   Note: When entering these values for your Tomcat service, the values cannot contain variables. Any variables containing other variables must be

evaluated to absolute paths. For example, if `OTS_INSTALL=%OTS_HOME%/install/%OTS_VERSION%` then you must evaluate the variables in the string so that the value can be expressed as `OTS_INSTALL=C:\OpenText\OTS_Base\install\16.0.01_4168_160623`.

4. In the Apache Tomcat properties dialog for your instance, switch to the **Java** tab and in the **Java Options** field, delete all existing values.

5. In the **Java Options** field, you must set the JVM parameters by copying and pasting the following values. Ensure that only one parameter appears per line and replace all variables indicated with the angled brackets with information about your Tomcat server and the values extracted from the `setenv.bat` file.

```
-XX:MaxPermSize=512m
-Xss1OM
-Dcatalina.home=<CATALINA_HOME>
-Dcatalina.base=<OTS_HOME>\TomcatBase\<instanceName>
-Djava.endorsed.dirs=<CATALINA_HOME>\endorsed
-Djava.io.tmpdir=<CATALINA_HOME>\temp
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djava.util.logging.config.file=<CATALINA_HOME>\conf\logging.properties
-Dots.mode=tomcat
-Dots.startup.mode=tomcat
-Dots.port=<ots.port>
-Dots.shutdown.port=<ots.shutdown.port>
-Dots.host=<ots.host>
-Dots.home=<OTS_HOME>
-Dots.install=<OTS_INSTALL>
-Dots.instance=<instanceName>
-Dsystem.componentdef.path=<OTS_INSTALL>/initialFiles/system
-Dots.war.path=<OTS_HOME>/install/<OTS_VERSION>/web
-Dots.infowebclient.path=<OTS_HOME>/web/tomcat/<OTS_VERSION>
-Dots.tomcat.loader=<OTS_HOME>/custom/jars/*.jar, <OTS_HOME>\TomcatBase
\<instanceName>/classpathFiles, <OTS_INSTALL>/lib/common/*.jar, <OTS_INSTALL>/lib/
endorsed/*.jar, <OTS_INSTALL>/lib/clientOnly/*.jar
```

6. In the **Maximum memory pool** field, enter the maximum amount of space in megabytes you want to allot for the memory pool.

7. When you are finished, click **OK** to save your changes and close the dialog.

When you are finished, proceed to "Step 3: Starting the Windows Service" on page 322.

6.7.4 Step 2B: Installing and Configuring the Tomcat Service (via Script)

Note: If you do not wish to use the script, the installation and configuration of the Tomcat service can also be completed manually. For more information, see "Step 2A: Installing and Configuring the Tomcat Service (Manually)" on page 320.

For each instance you created, you must establish a separate Tomcat service.

Navigate to the `<install_home>/TomcatBase/` directory where the enterprise application package and its related files are stored. Locate the `install-service-<instanceName>.bat` file and run it. (A separate file exists for each instance so users with multiple instances must ensure they are running the correct script file.) In a

command prompt window, the script automatically retrieves the required configuration settings from your instance and installs the new Windows service.

If installation of the service is successful, the following message appears on the command line:

Installed Apache Tomcat x OTS <instanceName> as service name TomcatOTS-<instanceName>

When you are finished, proceed to "Step 3: Starting the Windows Service" on page 322.

6.7.5 Step 3: Starting the Windows Service

With the service configured, you are now ready to start the service. Open the **Services** console in Windows and locate your newly created service in the list (for instance, `Tomcat9<instanceName>`). Select the service and click **Start Service**.

6.8 Switching Between Multiple Instances

Output Transformation Server provides the ability to create multiple deployed instances of the application, but only one instance can be initialized and managed in the user interface at a given time. Switching between instances requires shutting down the application and modifying some settings in the properties files to initialize another instance during the next restart.

To switch to another instance, navigate to your `<install_home>\startup` directory, locate the file you use to launch the product (either `startDesigner.bat/.sh` or `startEngine.bat/.sh`) and open the file for editing. Within the file, you must add an `-instance=` command, which indicates the instance to use upon start up. There are two ways of specifying the desired instance to initialize in the value:

- Instance name as designated while packaging the application.
- Instance number, as shown in the `startup.properties` file. (You must create the subsequent instances before their respective instance numbers are generated in the file.)

Note: Instead of hard coding the instance name or number and saving it in the file, you can choose to pass `-instance=` as an argument with the chosen instance specified when launching the batch file.

6.9 Stopping a Standalone Instance of the Engine

Every time you start a standalone instance of the engine using the `StartEngine.bat \.sh` file, a command file is created in the `\startup` directory named `CommandFile_<InstanceName>.txt`, where `<InstanceName>` represents the name of your engine's instance. If the command file already exists, the file is overwritten with subsequent engine start ups.

To stop a standalone instance of the engine:

1. Navigate to `<install_home>\startup` and locate the `CommandFile_<InstanceName>.txt` file for the engine instance you wish to stop. Open this file in your preferred source code editor for editing.
2. Determine the method for shutting down the engine and uncomment the line containing your selection. You can choose from:
   - **cmd=exitasap**. Aborts all running jobs immediately before shutting down the engine.
   - **cmd=flush**. Waits for all running jobs to end before shutting down the engine. Additionally, no new jobs are accepted while the engine waits for the currently running jobs to end.
3. Save your edits to the file.

   The command file is assessed every five seconds and it is only respected if the last modified time/date has changed. When the file is recognized by the system, your command is immediately run and the engine is shut down at the specified point. The engine's shutting down progress can be tracked in the command prompt window that opens when you start the engine. When it is finished, you are prompted to press any key to close the window.

Note: The process to stop the engine can be automated by creating a script to replace the file with an edited copy of the file containing your preferred engine stop command.

Chapter 7

OpenText Output Transformation Server Configuration Manager

The OpenText Output Transformation Server Configuration Manager is a centralized location that stores:

- **Process flows**. A process flow is several processes combined into one that can be executed as a series, in a specific order.
- **Project configurations**. Project configurations are the settings that determine how your project performs.
- **Resources**. A resource is something that is needed by a process flow or project configuration. This includes configuration files for components, fonts for Output Transformation Engine, and dictionaries for Data Transformation Engine.

Beyond its basic repository features, the Output Transformation Server Configuration Manager is also a comprehensive processing solution that includes functions to support collaborative efforts on the stored process flows, project configurations, and resources such as versioning, configuration creation and deletion, and check-ins and check-outs. Think of the Output Transformation Server Configuration Manager as a file cabinet with an administrator organizing the files within it. The person can add, remove, use, replace, and edit the files contained in the filing system. That individual can also share files between file cabinets. That is fundamentally what the Output Transformation Server Configuration Manager does; it works with the contents within the repository to ensure the maximum potential is reached.

Functional Summary

Once project configurations and resources are stored, they can be deployed easily across multiple systems for development, testing, staging, and final production. Use of the Output Transformation Server Configuration Manager will help to overcome the organizational issues that can arise from collaborative efforts as it ensures that everyone uses the most current iteration of the projects.

Configuration versioning with check-in/check-out functionality enables developers to collectively develop projects with common version control functionality, reducing the occurrence of errors, speeding deployment, minimizing internal administration, and enabling the transfer of skills among employees. Versioning also assigns an incremental number that corresponds to new developments in the project files, and allows for rollbacks when necessary.

Grouping individual configurations into a solution enables rapid deployment across multiple systems throughout the project lifecycle.

Target Users

The versatility of Output Transformation Server coupled with the Output Transformation Server Configuration Manager eases the organizational hurdles of working in partnership within a team or and even between companies. The Output Transformation Server Configuration Manager will benefit users who are running several servers in a cluster, have multiple people working on the same project, need a centralized repository along with versioning for project files, and want to move projects easily from one environment to another.

7.1 Configuring User Access and Output Transformation Server Configuration Manager Options

In Output Transformation Server, there are tasks that you can perform to control access to and manage resources for the Output Transformation Server Configuration Manager.

7.1.1 Allowing User Access

Besides the standard menu options available through Output Transformation Designer, there are three applications or tools that allow you to control user and administrator access to the Output Transformation Server Configuration Manager:

- **Output Transformation Server Manager.** For more information, see *OpenText Output Transformation Server Manager User Guide*.
- **Database application**. For more information, consult your database application online help.
- **Environment Manager**. For more information, see "Configuration Manager Administration" on page 337.

7.1.2 Managing Resources Within the Output Transformation Server Configuration Manager

To work with resources within the Output Transformation Server Configuration Manager, right-click a resource on a mounted file system and select one of the following Output Transformation Server Configuration Manager options:

- "Creating a Solution" on page 330
- "Adding a Solution" on page 330
- "Labelling a Solution" on page 331
- "Dropping a Solution" on page 331
- "Adding Resources" on page 332
- "Dropping Resources" on page 333

7.2 Setting Up the Output Transformation Server Configuration Manager

To set up Output Transformation Server Configuration Manager:

1. Ensure a database application is properly configured and running on your machine.
2. In Output Transformation Designer, start the Package and Deploy Wizard and follow the steps in the Package and Deploy Wizard to:
   - select an application server
   - set up the Base and Configuration Repositories
   - connect to a database for Configuration Manager (you may use an existing database or opt to create a new one)
   - package and deploy an EAR (bypass this step if your setup is on a standalone Output Transformation Server)
   - start Output Transformation Server from an application server

   Note: For more information about the Package and Deploy Wizard, see "Setting up a Server" on page 293.
3. Set up Output Transformation Designer to communicate with Configuration Manager, which must be running during server setup to enable Output Transformation Designer to communicate with it.
   a. In Output Transformation Designer, navigate to **Tools** > **Configuration Manager**.
   b. In the **Configuration Manager** dialog box, select a configuration set from the left pane.

      If this is the first time you are setting up **Configuration Manager**, a default configuration set named **New Configuration** is selected for you to arra. If you prefer, you can create a new configuration set to use by clicking .
   c. Once you have selected the configuration set to use, select the **Enabled** check box to turn on the Configuration Manager feature and allow configuration options to be set.

d. Next, you must complete the following information about the remote server:

- **Name**. Specifies the name used to identify the configuration set.
- **Host**. Specifies the host server IP address.
- **Use HTTPS**. Indicates that HTTPS is used on the application server and access through this connection is made through the HTTPS protocol.
- **Port**. Specifies the port to connect on.
- **Configuration Manager User**. Specifies the administrator's sign in name for the connection.
- **Configuration Manager Password**. Specifies the administrator's associated sign in password.

Note: For reference, the administrator sign in name and password used here must correspond with the respective `registryUser` and `registryPassword` values in the `XenosEngine.config` file.

e. Click **Test** to ensure the connection to the remote repository is functioning.

A successful test means Output Transformation Designer is set up to communicate with the Configuration Manager. If the test fails, no dialog box appears and then you should ensure that the server is running or check your configuration settings.

f. When you are finished, click **Activate** to initialize the instance of Configuration Manager and sync your resources with the database.

Note: In addition to the Configuration Manager options available within Output Transformation Designer, there are some supplementary options available on the **Configuration Manager** tab in Output Transformation Server Manager. For more information, see Output Transformation Server Manager User Guide*OpenText Output Transformation Server Manager - User Guide (VDTOTS-H-USM)*.

Related Links

- "Setting up a Server" on page 293
- "Using the Output Transformation Server Configuration Manager" on page 329

7.3 Using the Output Transformation Server Configuration Manager

The Output Transformation Server Configuration Manager is the centralized repository that stores your team's solutions, project configurations, process flows, and resources.

To work within the Output Transformation Server Configuration Manager, you need to:

- mount a file system before you work within the Output Transformation Server Configuration Manager
- have at least one solution under the file system's mount point
- add at least one resource to the solution

7.3.1 Mount a New File System

To mount a new file system:

1. In the **File Systems** window, right-click on **File Systems**, select **Mount** from the context menu, and then the type of file system you want to mount. You can select from either **Local Directory** or an **Archive**.

   The **Mount File System** dialog opens.

2. This will be the main work area for the engine and the location for all of the configuration files. Choose the directory where you want to mount the files, and click **Mount**.

   The folder appears in the **File Systems** window.

7.3.2 Solutions

A **solution** is a package of resources and project configuration files that are put in the Output Transformation Server Configuration Manager. The **Solution Manager** displays all the solutions currently deployed on the server.

7.3.2.1 Solution Manager

The **Solution Manager** displays all the solutions currently deployed on the server. To open the Solution Manager, right-click the top level of a mounted file system select **Solution Manager** from the context menu.

The options available in the Solution Manager are:

- **New.** Creates a new solution. For more information, see "Creating a Solution" on page 330.
- **Drop.** Permanently removes a solution from the Output Transformation Server Configuration Manager. You can also drop a solution from the mounted file system. For more information, see "Dropping a Solution" on page 331

- **Import.** Imports an existing solution to the Output Transformation Server Configuration Manager.
- **Export.** Exports a snapshot of the current configuration (the selected solution) from the Output Transformation Server Configuration Manager. You need to select a label the export target from the list in the Export dialog first to export the configuration.

7.3.2.2 Creating a Solution

A **solution** is a package of resources and project configuration files that is stored in the Output Transformation Server Configuration Manager.

To create a new solution:

1. In the **File Systems** window, right-click the top level of a mounted file system, and from the context menu, do one of the following:
   - Choose **Solution Manager**, and then click **New**.
   - Select **New > Solution**.

   The **New Solution** dialog appears.
2. In the **Solution** field, provide a name for your new solution, and optionally, you can include some brief notes in the **Description** field.
3. When complete, click **OK**.

   The new solution is uploaded to the server, and appears in the **File Systems** window under the selected mount point. The solution is ready to use in the Output Transformation Server Configuration Manager.

7.3.2.3 Adding a Solution

A user uploads a new solution to the server when the solution is ready to use in the Output Transformation Server Configuration Manager. Users working on other servers may access this new solution from the Output Transformation Server Configuration Manager by adding it to the mounted **File Systems** window on their server. A remote connection needs to exist between the servers and the Output Transformation Server Configuration Manager.

To add a solution:

1. In the **File Systems** window, right-click on a mounted file system, and select **Add Solution** from the context menu.

   The **Add Solution** dialog opens with a list of available solutions from other remote servers.
2. Select a solution from the list to add, and click **OK**.

   The solution appears in the **File Systems** window under the selected mount point, and no longer appears in the Add Solution dialog. It can now be used in the Output Transformation Server Configuration Manager.

Note: You can select and add multiple solutions at a time.

7.3.2.4 Restoring a Solution

To restore a dropped solution under the selected mount point, create a new solution with the same name as the dropped solution. Creating the new solution restores the solution and its stored local resources. For more information, see "Creating a Solution" on page 330.

7.3.2.5 Labelling a Solution

A **label** identifies the current configuration of a solution before you export it. A label acts as a checkpoint for project configuration files and resources you have packaged in a solution to help you track their status during testing, development, or production. A solution may have many labels identifying these checkpoints.

To label a solution:

1. In the **File Systems** window, right-click on a solution, and choose **Repository** > **Label Solution** from the context menu.

   The **Label Solution** dialog opens.

2. In the **Label** field, provide an appropriate name for your solution's label.

3. Add optional, brief notes in the **Description** field.

4. Click **OK**.

Note: For more information on exporting a solution, see "Solution Manager" on page 329

7.3.2.6 Dropping a Solution

A solution with packaged project configuration files and resources may be removed from the Output Transformation Server Configuration Manager when it is no longer needed. The decision to delete the entire solution from the Output Transformation Server Configuration Manager or all but the local copies of the solution depends on whether or not you want to use the local files at another time.

To drop a solution:

1. In the **File Systems** window, right-click on a solution, and choose **Repository** > **Drop Solution** from the context menu.

   The **Drop Solution** dialog appears.

2. You can either:

   - Select the **Delete Local Copy** check box if you want to remove all resources from Output Transformation Server Configuration Manager permanently.

- Leave the **Delete Local Copy** check box clear if you want to keep local copies of the solution. They will be stored in another directory until you want to restore them.

3. Click **Yes** to drop the solution.

   The solution is deleted from the Output Transformation Server Configuration Manager. If you kept a local copy of the solution, you can restore it later.

7.3.3 Resources

A solution is a package of resources and project configuration files that is stored in the Output Transformation Server Configuration Manager. A **resource** under a solution may be a process flow, a project configuration file, a component, or even a branch of a mounted file system. Once a resource is added to a solution, it can be deployed easily across multiple systems for development, testing, staging, and final production.

7.3.3.1 Adding Resources

To add a resource:

1. In the **File Systems** window, ensure there is a mounted file system with a solution under it. This needs to be in place to add a resource.
2. Right-click the component in the **File Systems** window under the solution's mount point and choose **Repository** > **Add Resource** from the context menu.

   The **Add Resource** dialog opens.
3. Specify when the resource will be available for use by changing the settings in the **Time of Availability** and **Date of Availability** fields.

   The default is for resources to be available immediately upon checking in.
4. Add a comment, if desired, about the resource.
5. Click **OK**. A clock on the component's icon indicates the component is a resource now.

Note: If the resource you want to use with the Output Transformation Server Configuration Manager is not under the solution yet, you will add it before you can use it. For example, you could add a new component by right-clicking the solution and selecting **New** > **Component** from the context menu. Once the new component appears under the solution, follow the steps above to add the new component as a resource.

7.3.3.2 Restoring a Local Resource

When you drop a resource, you can keep a local copy of its files to use later. Before you use the resource again, you need to restore its local copy.

To restore a dropped local resource, right-click the component (dropped resource) in the **File Systems** window and select **Repository** > **Add Resource** from the context menu.

To restore a local resource belonging to a dropped solution, create a new solution with the same name as the dropped solution in the **File Systems** window under the selected mount point. Creating the new solution restores the solution and its stored local resources.

Related Links

- "Dropping Resources" on page 333
- "Creating a Solution" on page 330

7.3.3.3 Dropping Resources

To delete a project configuration file or resource:

1. In the **File Systems** window, right-click a resource, and select **Repository** > **Drop Resource** from the context menu.

   The **Drop Resource** dialog opens.

2. You can either:

   - Select the **Delete Working File** check box if you want to remove all resources from the Output Transformation Server Configuration Manager permanently.
   - Leave the **Delete Working File** check box clear if you want to keep local copies of the resource. They will be stored in another directory until they are ready to be restored.

3. Click **Yes** to confirm the resource deletion.

   The resource is deleted from the Output Transformation Server Configuration Manager. If you decided to keep the working file, you may restore it later.

7.3.3.4 Checking Out Resources

Prior to making changes to a resource, you need to check it out or lock it first. When you check out a resource to work on, the Output Transformation Server Configuration Manager will resynchronize and then lock the file. That way, you will have the latest version to work with, and no other users can make changes to it. If you lock a resource only, the file does not resynchronize and you prevent others from using the resource while you work on it.

To check out a file:

In the **File Systems** window, right-click on the resource you want to work with, and choose **Repository** > **Check Out** from the context menu.

A green arrow appears over the resource's icon to show that the file is checked out.

Related Links

- "Locking a Resource File" on page 335
- "Resynchronizing Resources" on page 335

7.3.3.5 Checking In Resources

When you have finished making changes to a resource, check it in to preserve the changes with a new version number in the resource's history. For more information, see "Versioning" on page 336.

To check in a resource:

1. In the **File Systems** window, right-click on the resource you want to check in, and choose **Repository** > **Check In** from the context menu.

   The **Check In** dialog opens.

2. You can:

   - Specify when the resource will be available for use by changing the settings in the **Time of Availability** and **Date of Availability** fields. The default is for the resources to be available immediately upon checking in.
   - Describe the changes you made, or enter any other pertinent information in the **Comment** field. The comments will appear in the Comment box on the History dialog for the selected resource.
   - Select the **Major Version Update** check box to indicate key changes. Selecting this check box increments the first digit in the version number by one. For example, if the current version number is 1.1, the new version created will be 2.0.

3. When complete, click **OK**.

   If the application detects that no changes have been made to the file since it was checked out, you will be prompted to confirm the checking in of an unchanged resource.

7.3.3.6 Locking a Resource File

Locking a resource by itself prevents other users from using the resource while you work on it. The resource files are not resynchronized. When you check out a resource to work on, Configuration Manager resynchronizes and then locks the file. That way, you will have the latest version to work with, and no other users can make changes to it.

To lock a resource:

In the **File Systems** window, right-click on the resource you want to work with, and choose **Repository** > **Lock** from the context menu.

A green arrow appears over the resource's icon to designate that the file is locked.

Related Links

- "Resynchronizing Resources" on page 335
- "Checking Out Resources" on page 334

7.3.3.7 Unlocking a Resource File

To unlock a resource:

In the **File Systems** window, right-click on the resource you want to unlock, and choose **Repository** > **Unlock** from the context menu.

7.3.3.8 Resynchronizing Resources

With multiple users working within the same Configuration Manager instance, it is common for resources to become out of sync with the version on the server. For instance, other users' revisions may have led to a resource version of 2.8, while the version shown to you in Output Transformation Designer is still 2.3.

The Configuration Manager verifies that you have the latest version prior to allowing you to either check out or lock a file. If you do not have the latest version, a dialog appears to notify you.

To resynchronize your resources:

In the **File Systems** window, right-click on the resource, and choose **Repository** > **Resynchronize** from the context menu.

The resource is resynchronized to the latest version.

Related Links

- "Checking Out Resources" on page 334
- "Locking a Resource File" on page 335

7.3.3.9 Retrieving Resources Automatically

If there is a **direct** connection between the machine running Output Transformation Server and the Output Transformation Server Configuration Manager, Output Transformation Server retrieves checked-in solutions, solution updates, resources, and project configuration files automatically each time changes are checked in. This ensures multiple users have access to the most current versions of solutions or resources.

Output Transformation Server automatically searches the Base Repository first for solutions or resources before searching within Configuration Manager for them.

If:

- there are no solutions or resources in the Base Repository yet, Output Transformation Server searches the Output Transformation Server Configuration Manager and retrieves the resources automatically when it finds them.
- on startup, Output Transformation Server finds solutions in the Base Repository, it searches automatically for updates in the Output Transformation Server Configuration Manager's updated cache. This helps to limit the number of retrieved solutions at one time.

7.3.3.10 Versioning

The Output Transformation Server Configuration Manager is able to track the changes made to a component upon checking in. These changes are preserved with a new version number in the revision history.

By default, when checking in a file the version number increments by one. For example, if the current version is 1.6, the new version is designated 1.7. If you want to indicate that more radical changes have been made to a resource, on the **Check In** dialog, select the **Major Version Update** check box; this increments the first digit in the version number by one. For example, if the current version is 2.1, the new version created is 3.0.

7.3.3.11 Viewing the Version History of Resources

Output Transformation Server Configuration Manager tracks the progression of changes to a resource. For any version, you can view:

- which user was responsible for the changes
- which user locked the resource
- when the resource was checked out
- when the resource is scheduled to launch
- comments written when the resource was checked in

To view the history of a resource:

1. In the **File Systems** window, right-click on the resource, and choose **Repository** > **Show History** from the context menu.

   The **History** dialog opens. A green arrow appears next to the version you are viewing.

2. You have the following options to enhance your view of the information:

   - **Set the version to view**. Right-click the version you want to view the history for and select **Set Version**. The green arrow moves to that version, which becomes the current version.
   - **Compare version differences**. Right-click the version you want to compare differences with and select **Differences**. (You will need a third-party GUI-based version comparison tool, like UltraCompare or WinDiff, to see the differences between the versions.)
   - **Filter the number of columns in your view**. Click the **Ellipsis**,  , at the bottom right of **Version History**, select the columns to view, and click **OK** to change your view.

3. Click **OK** to exit the **History** dialog box.

   The default view is restored.

7.4 Configuration Manager Administration

Individuals need a user name and password set up by an administrator to use Configuration Manager.

7.4.1 Types of Users

Different types of users have various permissions levels for access:

- **Administrator**. The administrator is responsible for maintaining the operation of Configuration Manager. The administrator sets up Configuration Manager and looks after users' access privileges.
- **User**. Users can access the solutions and resources in Configuration Manager.

7.4.2 User Setup

To add a new administrator or user to the Output Transformation Server Configuration Manager, use the **Environment Manager**.

1. After the Output Transformation Server Configuration Manager has been set up successfully, choose **Environment** > **User Administration**.

   The **User Administration** dialog opens.

2. Click **Add**.

   The **Add User** dialog opens.

3. Complete the **Login** and **Password** fields. The **First Name**, **Last Name**, and **Email** fields are optional.
4. Choose Administrator or **User** from the **Role** drop-down menu.
5. Click **OK**. A new administrator or user is added.

Tip: There are two other tools you can use to add a new administrator or user for Configuration Manager: in either Output Transformation Server Manager and the database application used with the Configuration Manager. Consult *OpenText Output Transformation Server Manager - User Guide (VDTOTS-H-USM)* or the database application's documentation to guide you through the setup process.

Chapter 8

Error Handling

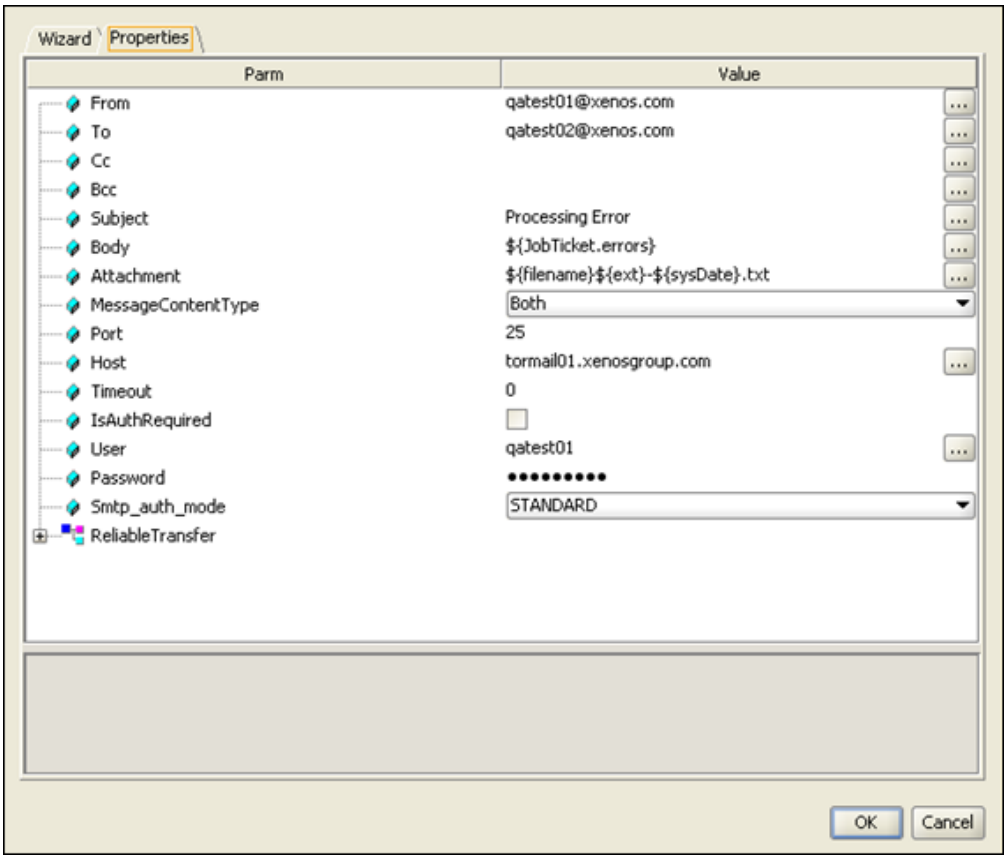
8.1 Using Error Variables

When an exception is thrown within a process flow, an error variable will be set to the stack trace that has occurred. The variable can then be used in another process or event, such as the Mail Process, which would email the contents of the error message to a predefined address. This is typically done in the process or process flow attached to the Catch on Error icon. For more information, see "Catch onError" on page 69.

The variables are:

- **JobTicket.warnings**. Usually trigger non-fatal errors. For example, if a file process is configured to write to a specific file that already exists and the writeMode is set to do not overwrite, the warning would be that the file already exists and the non-fatal error would be that the file process did not complete successfully.
- **JobTicket.errors**. Indicate conditions that lead to an immediate end of a job where the Output Transformation Server engine does not fail.
- **JobTicket.fatals**. Indicate absolute error conditions where the Output Transformation Server application fails.
- **JobTicket.humanerrors**. Indicate errors such as invalid file paths, invalid file names, or other non-program errors, created by user input.

The following shows a JobTicket.errors variable used in a mail process:


Wizard
Properties
Parm
Value
From
qatest01@xenos.com
To
qatest02@xenos.com
Cc
Bcc
Subject
Processing Error
Body
${JobTicket.errors}
Attachment
${filename}${ext}-${sysDate}.txt
MessageContentType
Both
Port
25
Host
tormail01.xenosgroup.com
Timeout
0
IsAuthRequired
User
qatest01
Password
Smtp_auth_mode
STANDARD
ReliableTransfer
OK
Cancel

Chapter 9

Known Issues and Limitations

This section lists the known limitations that have been identified in Output Transformation Server. Wherever possible, workarounds are provided.

9.1 General Issues

Some database tables cannot be created over a JNDI connection when Output Transformation Server is running outside the application server containing the JNDI resources.

If you are trying to create a JobAudit or JobStats table with Output Transformation Server when working on an application server via a JNDI connection and you have Output Transformation Server running outside of the application server that contains the JNDI resources, the new database tables are not generated. Consequently, no job statistics can be written to the tables.

To workaround this issue, at the outset connect to your database server with a JDBC connection to initially create the JobAudit or JobStats tables. Then, after you have successfully created the tables with Output Transformation Server, you can change your connection type back to JNDI since job statistics can be written to existing tables through JNDI with no issues.

9.2 CMIS Adapter known issues and limitations

Projects using the CMIS Adapter throw a ClassCastException and cannot be executed successfully.

To workaround this issue, remove the `jaxb-api-2.1.jar` and `jaxb-impl-2.1.7.jar` files from the `Axis2.war` file located in the `<install_home>\install\<version>\web` folder. After the jar files are removed, you must rerun the deployment packaging scripts to repackage the corrected files.

9.3 HTTP Event known issues and limitations

Projects using the HTTP Event component fail when run in Java 11 environments.

The failure is due to DSA certificates not being supported by TLS v1.3. To workaround this issue, you can manually set the protocol version to 1.2, which supports DSA certificates, by setting the following system property:

```
-Djdk.tls.server.protocols=TLSv1.2
```

9.4 JBoss Web Server known issues and limitations

JBoss Web Server users can successfully deploy a deployment package to their application server, however, cannot log into Output Transformation Server Manager due to an URI is not absolute error message.

When a JBoss Web Server user attempts to log in to Output Transformation Server Manager, the following message is recorded in the log file:

```
java.lang.Exception: No values found.
     at
com.xenos.admin.server.AdminServiceConnection.checkValue(AdminServiceConnection.java:
90)
     at
com.xenos.admin.server.AdminServiceConnection.initialize(AdminServiceConnection.java:48)
```

The problem is due to the instance's properties file containing the incorrect name. To workaround this issue, navigate to the `<install_home>\settings` folder and manually rename the `appserver_<instance_name>.properties` file to `tomcat_<instance_name>.properties`.

9.5 Modifying the JDBC URL for Microsoft SQL Server users

Microsoft SQL Server users with connections created in a previous version of Output Transformation Server must update their JDBC URL.

Output Transformation Server version 23.4 and later contains Microsoft JDBC Driver for SQL Server with package version 12.2 because the previous driver version used by the application is now obsolete. The latest version of the drivers use encryption by default and if your server connection is not set up for SSL, you must modify your URLs accordingly.

When the JDBC connection URL does not include the required security protocols after upgrading Output Transformation Server, Configuration Manager users may encounter an error similar to the following:

```
Failed to initialize repository manager
...
Caused by: com.xenos.registry.RegistryException: Cannot create PoolableConnectionFactory

(The driver could not establish a secure connection to SQL Server by using Secure
Sockets Layer (SSL) encryption
```

To update the JDBC URL, if you are using Configuration Manager then you can run the Package and Deploy Wizard or the `SetupCrm.bat/.sh` script to update the URL.

You can also manually modify the connection URL. In the previous version of the application, the URL would be similar to:

```
jdbc:sqlserver://<hostname>:<port>;SelectMethod=cursor;DatabaseName=cm
```

In the latest version of the application, you must add the **encrypt** flag to their connection URL. The new URL will be similar to:

```
jdbc:sqlserver://<hostname>;<port>;encrypt=false;SelectMethod=cursor;DatabaseName=cm
```

The new encrypt flag also applies to connection URLs for Alternate Text Manager, Remediation Console, Data Transformation Engine, and Persistent Storage.

For more information about support for Microsoft JDBC Drivers for SQL Server and lifecycle, see the Microsoft website.

9.6 When trying to connect to the server from Output Transformation Designer, I am getting server connection issues.

To troubleshoot this issue, check the following settings:

- Verify that your host, port, username, and password are correct.
- When attempting to connect to the server in Output Transformation Designer, make sure the context root is set.
  - In the **Connect** dialog box, go to the **Advanced** tab to verify the **Context Root** value. The context root must match the URL used to connect to Output Transformation Server Manager via a web browser.