



Precisely EnterWorks

EnterWorks Workflow Activities and Callout BIC Framework

Version 10.4.8



Notices

Copyright 2007, 2022 Precisely.

Trademarks

"EnterWorks" and the "EnterWorks" logo are registered trademarks and "Enable PIM", "EnterWorks PIM", "EnterWorks Process Exchange" and "EnterWorks Product Information Management" are trademarks of Precisely.

Third-party Acknowledgments

Windows, .NET, IIS, SQL Server, Word, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Java and all Sun and Java based trademarks are trademarks or registered trademarks of the Oracle Corporation in the United States and other countries.

Oracle is a registered trademark and Oracle 10g is a trademark of Oracle Corporation.

Pentium is a registered trademark of Intel Corporation in the United States and other countries.

JBoss is a registered trademark of Red Hat, Inc.

All other trademarks and registered trademarks are the property of their respective holders.

All icons and graphics, with the exception of the "e." logo, were obtained from West Coast Icons and Design at <http://www.bywestcoast.com>.

Table of Contents

Overview	8
EPX Core Product Activities.....	8
Manual Activity	8
Automatic Activity.....	10
Automatic Activity – Audit BIC.....	11
Automatic Activity – Command Line BIC	11
Configuring the Command Line BIC	12
Creating a Script File for the Command Line BIC	13
Notes on Creating Command Line BIC Script Files.....	14
Using the Command Line BIC in a Process Flow	14
Using the Command Line BIC Editor	15
General Tab	16
Details Tab.....	17
Expiration Tab	19
Settings Tab.....	21
Custom Action Tab	24
Automatic Activity – E-mail BIC	26
Configuring the E-mail BIC	26
Inbound Tab	33
Outbound Tab	35
Automatic Activity – Epim Enable Handler BIC.....	37
Automatic Activity – EPIM Manage Item BIC.....	37
Automatic Activity – EPIM Manage Process BIC.....	37
Automatic Activity – File Attachment BIC.....	38
Automatic Activity – Flow Correlation BIC.....	41
Automatic Activity – FTP BIC.....	41
Automatic Activity – HTTP BIC	45
Settings Tab.....	45
Automatic Activity – JDBC BIC.....	48
Settings Tab.....	48
Configuring the SQL Option	49
Configuring the Stored Procedure Option	51
Automatic Activity – Load Properties BIC	54
Automatic Activity – Process Properties BIC.....	56

Automatic Activity – Purge Completed Workitems BIC.....	56
Settings Tab.....	56
Automatic Activity – Scheduler BIC.....	57
Using the Scheduler BIC as a Starting Point.....	58
Using the Scheduler BIC in the Body of a Process Flow.....	58
Automatic Activity – SOAP Request BIC.....	61
Settings Tab.....	62
Automatic Activity – Telnet BIC	62
Prompt Tab	63
Settings Tab.....	63
Automatic Activity – Universal Groovy BIC.....	65
Automatic Activity – Universal Java BIC.....	65
Automatic Activity – Universal Javascript BIC.....	66
Settings Tab.....	66
Automatic Activity – Universal Pimql BIC	69
Automatic Activity – Work Item Import and Export BIC.....	69
Work Item Import/Export BIC Format Options.....	69
The ImportExportAsynchBic.config File	72
Settings Tab.....	73
Import Tab.....	74
Export Tab	86
Delimited.....	87
Split Activity	100
Join Activity	100
Decision Point Activity	101
Finding Information for the DPA.....	102
Configuring DPAs.....	102
Configuring Rules for DPAs	103
Setting an Otherwise Rule for DPAs.....	105
Setting Advanced Rules for DPAs.....	106
General Guidelines for Establishing Rules	107
Valid Formats	107
Variables.....	108
Operators	108
Usage.....	109

Iterator Activity	110
Configuring Iterator Activities.....	110
Using Iterator Activities	111
Merge Work Items Activity	112
Work Item Priority Activity	112
Load Balancing Activity	112
Subflow Activity	113
Personal Subflow Activity	113
Remote Subflow Activity.....	113
End Activity	113
Callout BIC Framework	113
Callout BIC Interface	116
String getDescription()	116
void setInputParameters()	116
void setOutputParameters()	116
boolean processWorkItem(Work work, HashMap inputProperties, HashMap outputParameters, StringBuffer errors)	116
Callout BIC Methods	116
void addInputParameter(String name, String description, String[] options, String codeSetName, String defaultValue, boolean isRequired, int maxValues)	116
void addOutputParameter(String name, String description)	117
void cleanup()	117
void closeInput(BufferedReader br)	117
void closeOutput(PrintWriter pw)	117
boolean doesFileExist(String directoryName, String filename)	117
DBQuery epxGetQuery()	117
void epxFreeQuery(DBQuery dbQuery).....	118
void fatalError(String message)	118
void freeQuery(DBQuery dbQuery)	118
String getParameter(HashMap inputProperties, String name, String defaultValue)	118
String getSystemProperty(String propertyName)	118
DBQuery getQuery()	119
void logDebug(String channel, String message)	119
void logError(String message).....	119
BufferedReader newInput(String directoryName, String fileName, String charset)	119

PrintWriter newOutput(String directoryName, String fileName, String charset)	120
Example Callout Class	120
Registering and Configuring a Callout BIC	122
Creating Standalone Callout BIC	126
Callout BIC Modules	132
Advance Work Item BIC	132
Amazon S3 BIC	133
Attach Files BIC	136
CreateWorkItem BIC	137
DAM Upload and Link BIC	138
Delete Repository Record BIC	138
Export Repo Data Using Pref BIC	138
Flow Monitor BIC	139
Initiate Scheduled Export BIC	141
Initiate Scheduled Import BIC	142
Lock Export Data BIC	142
Lock Import Data BIC	143
Lock Workflow BIC	143
Missing Variants Report BIC	144
Promote Repository Record BIC	144
Regenerate Variants BIC	145
RepoDataCopyBic	146
RepoDataEmailBic	146
RepoDataUpdateBic	147
RepoTargetDataUpdateBic	147
SendEmail Callout	148
Send Rabbit MQ BIC	149
Set Properties BIC	149
Split Property BIC	149
Split Work Item BIC	150
SQL Copy Repository Records BIC	151
SQL Create Saved Set BIC	152
SQL Create Work Item BIC	153
SQL Repo Data Copy BIC	154
SQL Send Email BIC	155
SQL Set Properties BIC	156
SQL Update Repository Records BIC	156
SQL Workflow Notification BIC	157
Sync EPX Users with Enable BIC	158

EnterWorks Workflow Activities and Callout BIC Framework

Unlock Repository Record BIC	158
UnlockWorkflowBic.....	159
Update Repository Record BIC.....	159
Update User Group Assignments BIC	160
Validate Repository Record BIC	160
Workflow Notification BIC	161

Enterworks Workflows – Callout Framework and Automatic Activities

Overview

This document provides details on EPX workflow activities, including:

- Core Activities – details on the activities that are part of the EPX core product
- Callout BIC Framework – details on how to implement automatic activities using the Callout BIC Framework
- Callout BIC Framework Activities – details on the general-purpose activities built on the Callout BIC Framework and part of the Services Framework

EPX Core Product Activities

This section provides details on the EPX Core Product activities.

Manual Activity



The Manual Activity is used to launch work items on a workflow (either via Save and Send or an automated step using the CreateWorkItem BIC or SQL Create Work Item BIC. It is also used for any step in the workflow requiring manual intervention by a user.

The Details tab in the Manual Activity must identify the user or role (recommended) that is responsible for initiated the work item or responding to an active work item:

Manual Activity - Start

Manual Activity
Configure the basic properties of this activity

General Details Viewers Expiration Custom Action

Actor: EnableWorkflowInitiators

Activity Options

☐ Work Item Name Editable

☒ Starting Point

Default Work Item Name: Review

Work Item Priority: None

Persist Properties: Default

☐ Delayed Send

Work Item Key:

Date Format: MM-dd-yyyy HH:mm

Format Test: 04-08-2018 22:28

Enable PIM Options

☒ Linked to Enable PIM

Repository Name: SKU_Staging

Repository Friendly Name: SKU

Repository Id: 10016

Preference Name:

Preference Id:

Saved Set Name:

Saved Set Id:

Search Name:

Search Id:

Additional Search Attribute:

Item Id(s):

Task Name: Initiate Review

Task Instructions: Submit this record via Save and Send to launch a review on the record

Task Role:

Task Status:

Task Icon:

Task Notification: %workflowCommentHistory%

Task Object:

Allow Reassign: No Reassignment

Allow Listing Send: Do Not Allow Send

Listing Viewer Type: Editor

Work Item Type: StandardSubmission

Task Attributes JSON: [{"default": "Approved", "displayname": "Submission Type", "name": "workfl", "mission Notes", "name": "workflowComment", "type": "2", "list": []}]


If the work item is anchored to a repository record, then editing that record in Enable will provide access to the Manual Activity:

Edit 1 record(s)

Record 2 of 201004

Action
Reports

SKU



15110013
New Item

Error Levels

Review Record - Best Practice - Closed Loop

Approval

Task:
Review Record - Best Practice - Closed Loop Approval

Description:
Detailed instructions for the reviewer goes here.

Role:
Brian Zupke

Summary
Category Attributes
Marketing
Pricing
Item

Identifier

RH Retail SKU

15110013

Vendor Name

Ashley

CWLINK

15110013CRM

Non-Inventory (Y/N)

F -- [False]

UPC Code

VSN

123

Item Number

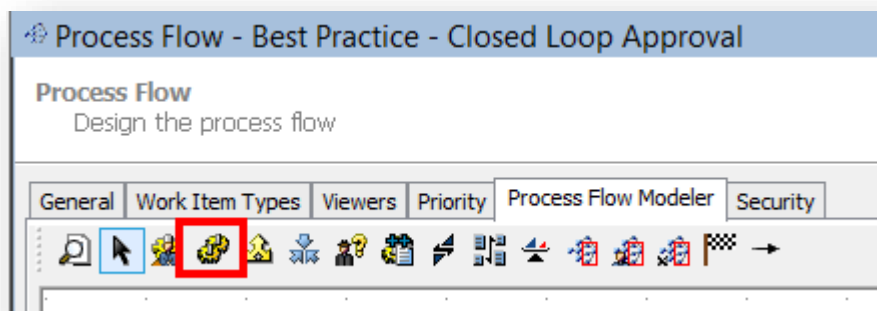
1410065450

Marketing Descriptions

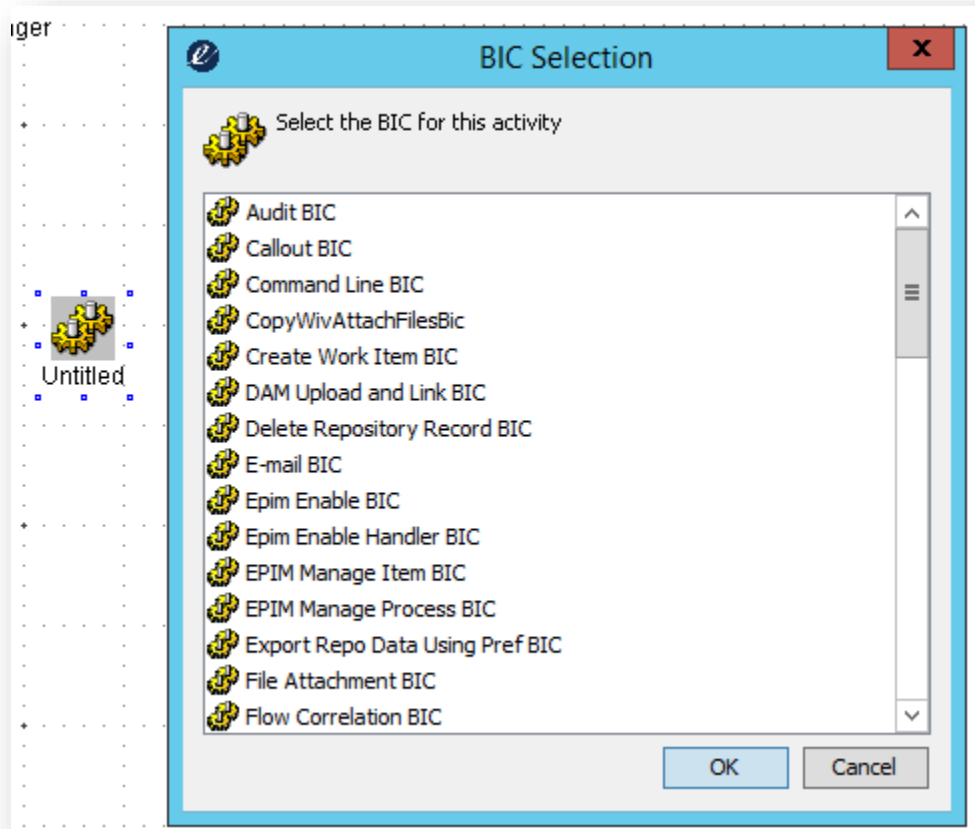
Automatic Activity



The EPX Design Console Flow Modeler Tab has a toolbar that lists the activities that can be added to the workflow canvas. One of these activities is the Automatic Activity:



When the Automatic Activity is selected and dropped onto the canvas, a selection prompt appears allowing you to select which automatic activity is to be used:



The majority of the Automatic Activities have unique icons but defaults to the double-gear if a unique icon has not been defined.

The following sections describe all of the EPX product automatic activities.

Automatic Activity – Audit BIC



The Audit BIC is a debugging activity that allows you to record the contents of the work item properties to the S_AUDIT table. It does not have any practical purpose for business process type workflows and is generally not used.

Automatic Activity – Command Line BIC



The Command Line BIC permits you to execute a script at the operating system command line or shell, and provides the ability to execute commands on the system where the BIC's associated BIC Manager resides. When implementing the Command Line BIC to modify a work item's hash table, you must specify the script file (which must already exist) and the hash table keys of work item to be used. The hash table is not updated directly when the Command Line BIC executes a script, but rather the script must produce a file that can then be loaded with the Work Item Import/Export BIC, at which point the work item's hash table is modified. (See the Work Item Import/Export BIC guide for more information.).

Note: You can use the Command Line BIC in any process flow or subflow. This BIC, however, cannot be a starting point.

Configuring the Command Line BIC

To optimize the Command Line BIC for your particular needs and system configuration, you need to edit the `CommandLineBic.config` file, installed on the BIC Manager host system in `<EPX>\bin\agents\<hostname>`.

The code sample below displays the default `CommandLineBic.config` file contents for a Windows installation:

```
# Available threads
minThreads=1
maxThreads=100
# command line run-time, uncomment the appropriate
# settings depending on the installed operating system
# Example configurations below
# Windows
command.shell.windows=cmd.exe
shell.args.windows=/c
# UNIX
command.shell.unix=/bin/sh
shell.args.unix=-c
```

Thread Configuration

The `minThreads` and `maxThreads` parameters are used to control the number of concurrent threads. As threads are created, they are added to a pool. The `minThreads` and `maxThreads` settings determine the size of this pool.

- `minThreads` is the minimum number of instances of the BIC that will be created. The lower the `minThreads` value, the more time will be needed for the BIC to start a new instance. By increasing the `minThreads` value, memory usage is increased, but processing time is decreased when a BIC instance is required. A value of zero (0), means that the thread pool can become empty when no work is done. The valid values are from 0 to 32767, with 0 as the default value.
- `maxThreads` is the maximum number of instances of the BIC that will be created. By increasing the `maxThreads` value, the maximum number of concurrent BIC instances is increased, denoting

that a higher volume of work can be passed to this BIC. Increasing the number too high could cause the BIC to consume the system resources on the server or servers hosting the BIC Manager and the instance of EPX's Application Server being used. The maxThreads should have a value of at least one (1) so that at least a thread can be created to process work items. The valid values are from 1 to 32767, with 32767 as the default value.

If minThreads is greater than maxThreads, their values will automatically be swapped and a warning will be displayed in bic.log.

Shell Configuration

The appropriate command.shell and shell.args settings for your operating system are activated by the EPX installation program. The code sample above shows shell settings for a Windows installation.

Creating a Script File for the Command Line BIC

Before you can use the Command Line BIC in a process flow, you must create a script file for the BIC to execute. Using a text editor, open a new file. You can enter any valid commands that are supported by the operating system in which EPX is running.

The Windows example below uses a script file and two batch files. As a customer's name is passed through the Command Line BIC activity, their name is placed in a file with a unique filename (the customer's name). A directory bearing the customer's name is then created and the file containing the customer's name is moved to the new directory. For the following example, the customer's name is Mike Johnson.

The sample script file contains the commands:

```
echo SET
CUSTOMER=%LastName%_%FirstName%>c:\docs\customer.bat
echo %FirstName% %LastName% > c:\docs\customer.txt
copy c:\docs\customer.txt
"d:\data\customers\%LastName%_%FirstName%.txt"
c:\docs\changedir.bat
```

- The script file first creates a batch file in c:\docs named customer.bat, which will be called in a moment. The batch file will be used to create a system variable called CUSTOMER, which will represent the customer's name in the format LastName_FirstName. (Executing a SET command directly from the script does not create a system variable.) For customer Mike Johnson, the batch file contains the command SET CUSTOMER=Johnson_Mike.
- The batch file then writes the name "Mike Johnson" to the file c:\docs\customer.txt. This file is for temporary use and is overwritten every time the Command Line BIC activity passes a new customer name.
- The script copies the contents of customer.txt as a new file, d:\data\customers\Johnson_Mike.txt. (Here, d:\data\customers is a pre-existing directory.)
- The script file then executes the batch file c:\docs\changedir.bat.

The task performed in this example now requires a directory change. However, changing directories is not allowed in scripts executed by the Command Line BIC, so the command to change directories is executed via a batch file (as are the remainder of the commands in this task).

The sample batch file, `changedir.bat`, contains the commands:

```
CALL c:\docs\customer.bat
d:
cd data\customers
md "%CUSTOMER%"
move "%CUSTOMER%.txt" ".\%CUSTOMER%\%CUSTOMER%.txt"
```

- The `changedir.bat` file first calls the `customer.bat` file that was created in the script. The `customer.bat` file executes, creating a system variable called `CUSTOMER` with a value of "Johnson_Mike".
- The `changedir.bat` changes the current drive to `d:`, and then changes the current directory to `d:\data\customers`.
- The batch file now creates a new directory under `d:\data\customers`, called `\Johnson_Mike`.
- Then, the batch file moves the file `Johnson_Mike.txt` from `d:\data\customers` (the destination to where it was copied by the script) to `d:\data\customers\Johnson_Mike`.

Notes on Creating Command Line BIC Script Files

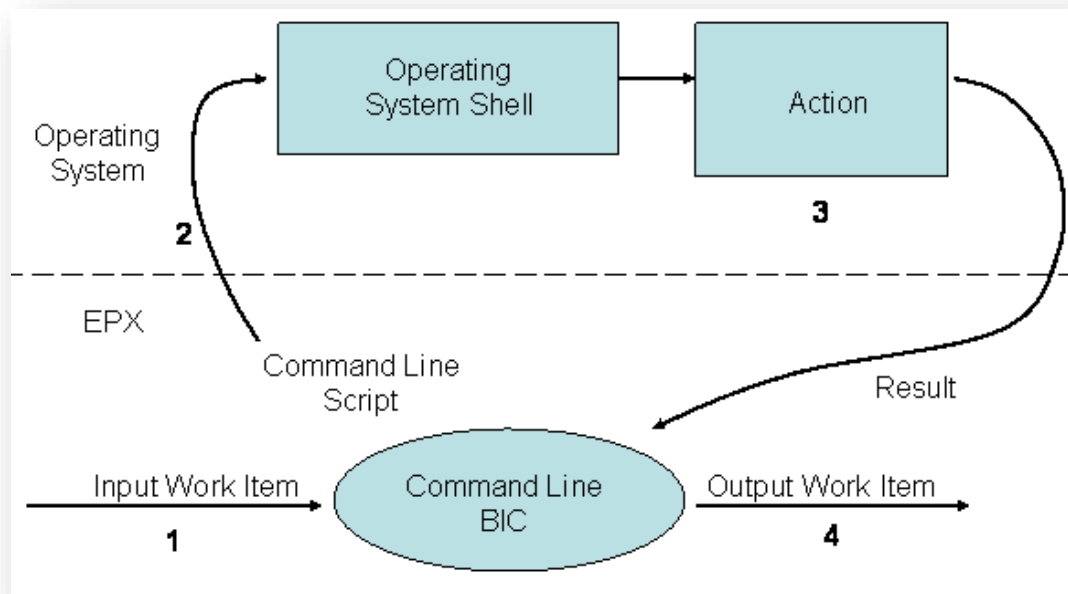
Before writing script files for use by the Command Line BIC, you should be aware of the following:

- The local directory is always the directory of the BIC Manager's application server. If EPX is installed to the default Windows directory, then the local directory is `C:\Enterworks\EPX\tomcat`.
- If you want to use existing system variables in the script file, you must prefix the percent sign (%) with a caret (^) — e.g., `^%TEMP^%` — as the script file uses the percent sign to define internal parameters. Also, as indicated in the earlier script example, you cannot define a system variable from within a script.
- You cannot change directories from within a script. Any commands to change directories must be executed from a batch file, as shown in the previous example.
- For directory names and file names that include spaces and more than eight characters, you must enclose the path with opening and closing quotes. See the sample string and batch file contents for examples of this.
- If your script writes a file that you want to preserve, you must include a string giving the new file a unique name. Otherwise, each time the process flow is started, the previous file will be overwritten. You can use any replaceable parameter that returns a unique value. You can also have the option for EPX to create a unique name for each file by using the system parameter:



```
%sys.workitem.version.uid%
```

Using the Command Line BIC in a Process Flow

The following diagram illustrates how the Command Line BIC acts in a process flow. (The numbers denote the order in which events occur.)



To configure a Command Line BIC activity in a flow, open the flow editor to the Process Flow Modeler tab and proceed as follows:

1. Click the activity icon  to place an automatic activity on the flow canvas.
2. Click the flow canvas and the BIC Selection dialog box appears.
3. Select Command Line BIC and then click **OK**.
4. Double-click the Command Line BIC activity icon  to open the Command Line BIC editor.
You can also right-click the HTTP BIC activity icon and then click Properties in the shortcut menu.

Using the Command Line BIC Editor

The editor's tabs are described in the following table.

Tab Name	Description of functions
General	Assign a name (required) and a description (optional) for the activity.
Details	Set a new work item priority if needed. Configure the BIC to send the work item to the next activity in the flow in the event that the BIC encounters an error. Specify which BIC Manager should be used, if the default is not sufficient. Indicate how the BIC should persist properties. Indicate the work item key that you will use to obtain the actual send date and/or time. Select or modify the format of the date by either choosing one from the list or entering an alternative.

Tab Name	Description of functions
Expiration	Specify whether the expiration should occur after a specific period, or whether the expiration should occur after a specific date and time. You can set the work item's expiration period by entering values, either by typing or by clicking the up and down arrows or indicate the work item key that you will use to obtain the actual send date and/or time. Select or modify the format of the date by either choosing one from the list or entering an alternative. An e-mail can be sent to a designated participant, or the work item can be sent on to the next activity.
Settings	Specify a script file for the BIC to use, and specify any properties from the hash table of work item to be used by the script file.
Custom Action	Indicate the fully qualified class name in the Send Action Class field to have a work item perform actions or tasks before sending the work item to the next activity. Specify additional property keys and values that you want to use during run-time.

General Tab

The General tab contains the following fields:

- Name – provide a name for the Command Line BIC activity. If you do not specify a name, EPX assigns a default name for the activity such as Automatic Activity (1) or Automatic Activity (3).
- Description – add other information on the BIC activity, if desired.

The screenshot shows a dialog box with five tabs: General, Details, Expiration, Settings, and Custom Action. The 'General' tab is selected. It contains a 'Name:' label followed by a text box containing 'Automatic Activity (3)'. Below this is a 'Description:' label followed by a large, empty text area. At the bottom of the dialog box are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

To save the data entered and proceed to another tab, click Apply. Clicking OK will also save the data entered and exit the Command Line BIC editor. To cancel saving the data entered, click Cancel.

Details Tab

The screenshot shows a software window with five tabs: General, Details, Expiration, Settings, and Custom Action. The 'Details' tab is active. It contains two main sections: 'Starting Point' and 'Delayed Send'. The 'Starting Point' section has a checkbox that is unchecked, followed by a text field 'Default Work Item Name' containing 'Default Work Item', a dropdown 'Work Item Priority' set to 'None', and another unchecked checkbox 'Send work item on error'. The 'Delayed Send' section has a checked checkbox, followed by a dropdown 'BIC Manager' set to 'System assigned', a dropdown 'Persist Properties' set to 'Default', a text field 'Work Item Key' containing 'DelayedSendDate', a dropdown 'Date Format' set to 'MM-dd-yyyy HH:mm', and a text field 'Format Test' containing '01-25-2003 18:05'. At the bottom right are four buttons: OK, Cancel, Apply, and Help.

The Details tab allows the user to set the following options:

- **Work Item Priority** – Specify which work item will be prioritized by selecting a work item from the Work Item Priority dropdown list. The new priority takes effect when a work item arrives at the Command Line BIC activity. The priority you select overrides any previously established priorities for work items. For an explanation of work item priorities and the available options, see the Process Modeling guide.

- Send work item on error – When the BIC encounters an error with a work item, you can have the BIC activity forward the work item to the next activity in the flow by enabling this option. Click the Send work item on error checkbox to enable this option.

Note: This option is typically used during the creation of a new process flow and allows users to test all activities in the flow for errors. If the option is not selected and an error occurs with the BIC, the work item will remain in the BIC's Inbox. To test a BIC activity using the Send work item on error option, you can use the Generic work items viewer for the activity following the BIC, at least until the BIC activity is functioning successfully. The Generic work item viewer allows you to view any error messages generated at the BIC activity. When the process flow is ready for actual use, deselect this option.

- BIC Manager – By default, the system determines which BIC Manager executes the BIC activity. You can override this by selecting a BIC Manager from the BIC Manager dropdown list.
- Persist Properties – The default persist properties setting for all activities in process flows running in the local iteration of EPX's Application Server is controlled on the Server editor's Details tab. To change the persist properties settings for the Command Line BIC activity, select a setting from the Persist Properties dropdown list. The settings are the following:

Default – Select this setting to enable the default persist properties setting for the local iteration of EPX Application Server.

No – Select this setting to turn off the persist properties function.

Yes – Select this setting to always persist properties.

To track changes to a work item as it passes from one actor to the next in a process flow, the properties of a work item version can be recorded or persisted in an actor's Sentbox.

- Delayed Send – Users can specify if the work item is not to be sent to the next activity immediately by clicking the Delayed Send checkbox. When you enable the Delayed Send option, you must specify the following:

Work Item Key – The key you will use to get the actual send date and/or time during runtime. In this example, type in "DelayedSend.Date" wherein DelayedSend is the work item type you created and Date is the string field in the work item type. You can also use the custom code in a class specified in the Send Action Class field on the Custom Action tab to set the delayed send time.

Date Format – The default format is MM dd yyyy HH:mm. Select a date format from the Date Format dropdown list or enter values in the combo box.

Format Test – This field displays the current date and/or time formatted using the format selected in the Date Format field. Each time a new format is selected, this value

is updated to reflect the newly selected format. Refer to the Appendix” on page 16 for the available date formats and time zones.

To save the data entered and proceed to another tab, click Apply. Clicking OK will also save the data entered and exit the Command Line BIC editor. To cancel saving the data entered, click Cancel.

Note: The Starting Point checkbox and Default Work Item Name field are disabled because the Command Line BIC cannot be set as a starting point.

Expiration Tab

The Expiration tab is used to specify how long a work item can remain at the Command Line BIC activity and what will happen to the work item if that period is exceeded. Setting an expiration period is optional, but should an error occur with the BIC Manager, an expired Command Line BIC activity will indicate that the BIC Manager was unavailable for a time longer than the expiration period.

To enable work item expiration:

1. Click the Expiration Period checkbox.

The screenshot shows a configuration dialog box with the 'Expiration' tab selected. The 'Expiration Period' section is checked. Under 'Specific Period', the 'Days' field is set to 0, 'Hours' is 0, and 'Minutes' is 0. The 'Dynamic Date' section is also visible, with a 'Work Item Key' field, a 'Date Format' dropdown set to 'MM-dd-yyyy HH:mm', and a 'Format Test' field showing '04-29-2003 08:33'. The 'Action' section contains two unchecked checkboxes: 'Send Work Item' and 'Send E-mail'. Below these is an 'E-mail' field with the text 'User/Group/Role' and a button to its right. At the bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

2. Specify when the expiration should occur by clicking any of these options:

Specific Period – If the expiration is set to Specific Period, set the amount of time until work item expiration by entering values in Days, Hours, Minutes, and Seconds fields. You can manually enter the values or click the up and down arrows to fill out the fields. Make sure that at least one of the four fields should be filled out.

The Seconds value must be at least 15 seconds. Setting an expiration period less than 60 seconds will cause delay in the actual work item expiration period because, by default, the Control Manager checks for expired work items every 60 seconds. The expiration polling interval can be reduced by modifying the value of the `control.expiredWorkItem.interval` setting in `<EPX>\bin\config.properties` (in the Control Manager Properties section), but before doing so you should consider the impact that this change will have on the performance of your system.

Dynamic Date - If the expiration is set to Dynamic Date, you will need to provide the following:

Work Item Key – The key you will use to get the actual send date and/or time during runtime. For example, type in "DynamicExpiration.Date" if the expiration date is designated by the Date field of the DynamicExpiration work item type in the work item.

Date Format – The default format is MM dd yyyy HH:mm. Select your desired date format from the Date Format dropdown list or enter values in the combo box.

Format Test – Contains the current date and/or time formatted using the format selected in the Date Format field. Each time a new format is selected, this value is updated to reflect the newly selected format. Refer to “Appendix,” on page 16 for the available date formats and time zones.

Instead of setting the expiration date by specifying a work item key, a send action class can be used to set that date. Click the Custom Action tab and enter the fully qualified class name for the custom code in the Send Action Class field.

After filling out the necessary fields, proceed to the next step.

3. Click the Send Work Item checkbox to automatically send the work item to the next activity in the flow upon expiration.

Note: The work item cannot go to another point in a flow that needs manual intervention to select a participant. For example, the work item can go to an All split to specify that all participants receive the work item following the split. If it is a Some or a One split, however, manual intervention is required to choose a participant.

4. Click the Send E-mail checkbox to automatically notify one or more participants, groups, or roles should the activity expire. When you select the Send E-mail checkbox, the E-mail table is enabled. Use the E-mail table to add e-mail recipients. To add e-mail recipients, select one or more recipients from the E-mail address selection dialog. For more information about the Expiration tab, see the Process Modeling guide.
5. Save the data entered and proceed to another tab by clicking Apply. Clicking OK will also save the data entered and exit the Command Line BIC editor. To cancel saving the data entered, click Cancel.

Settings Tab

The Settings tab contains the Script File Location and Result Set boxes and the property hash parameters table.

To add parameters to the property hash table:

1. In the Script File Location field, specify the path to the script file by clicking the Browse button and selecting the script file from the File dialog box or by typing the script file path.

Note: When the BIC is used in a clustered environment, make sure that the folders and/or script files that will be indicated are existing in and pointing to where the BIC Manager server is installed.

2. In the Result Set box, enter the name for the executed Command Line BIC activity's result set. A result set is a string that stores the output of the executed script. You would see this same output onscreen if you were manually executing the script in the operating system shell.

The result set name and its output string are a key/value pair. The result set string can be viewed by adding its key to a work item viewer.

Important: Result set names should not include spaces, or characters other than A-Z, a-z, 0-9, and the underscore character (_).

3. Click Add to enter parameter keys and values into the property hash parameters table. In the table cells that appear, type the name of the first key used by the script file, followed by its value. The value can be a Process Exchange work item variable or constant, a Process Exchange system variable or constant, or a constant that you choose yourself. Repeat this procedure for each key in the script file.

General | Details | Expiration | Settings | Custom Action

Script File Location: C:\Docs\namescript.txt Browse

Result Set: NameResults

Parameter Key	Parameter Value
FirstName	%Names.firstname%
LastName	%Names.lastname%

Add Remove

OK Cancel Apply Help

Note: If your script writes a file and you want to preserve each written file, you must include a string that gives each new file a unique name. Otherwise, each time the process flow is started, the previous file will be overwritten. You can use any replaceable parameter that returns a unique value. You can have Process Exchange create a unique name for each file by using the Process Exchange system parameter:

%sys.workitem.version.uid%

4. Save the data entered and proceed to another tab by clicking Apply. Clicking OK will also save the data entered and exit the Command Line BIC editor. To cancel saving the data entered, click Cancel.

Custom Action Tab

The Customs Action tab allows users to set the Delayed Send and Expiration Date by custom code in a specified class. You can use custom code to have a work item perform certain actions or tasks before sending the work item to the next activity. Refer to the section “Customizing Send Action Class,” found in the Process Modeling guide for detailed instructions on how to create a custom code.

General | Details | Expiration | Settings | Custom Action

Send Action Class:

Custom Properties

Key	Value
newUserLogin	jsmith

Open
Add
Delete

OK Cancel Apply Help

To set the Delayed Send and Expiration period by custom code:

1. Enter the fully qualified class name in the Send Action Class field.

The Send Action Class field is only available for the following EPX activities: Manual, Automatic, Anonymous, Iteration, Work Item Purge, and Decision Point. This would mean that the Send Action Class field can function independently and can be used without the Dynamic Date or Date Expiration feature.

Note: The Send Action Class field on the Custom Action tab or the Work Item Key field on the Details and Expiration tabs must have valid entries if Delayed Send or Dynamic Date is selected.

2. Specify additional property keys and values that you want to pass to the custom code identified by the Send Action Class or the Custom Expiration Handler, if one has been installed. You can add an entry to the list by right clicking anywhere in the key-value region and then clicking Add.
3. Type in the name and value of the activity in the Activity Property window.
4. Save the data entered and proceed to another tab by clicking Apply. Clicking OK will also save the data entered and exit the Command Line BIC editor. To cancel saving the data entered, click Cancel.

Date Formats

By including the following characters in the date format, you can specify a broad range of date formats:

Symbol	Meaning	Presentation	Example
G	era designator	(Text)	AB or BC
y	year	(Number)	1996
M	month in year	(Text & Number)	July & 07
d	day in month	(Number)	10
h	hour in am/pm (1 – 12)	(Number)	12
H	hour in a day (0 – 23)	(Number)	0
m	minute in hour	(Number)	30
s	second in minute	(Number)	55
S	Millisecond	(Number)	978
E	day in week	(Text)	Tuesday
D	day in year	(Number)	189
F	day of week in month	(Number)	27
w	week in year	(Number)	27
W	week in month	(Number)	2
a	am/pm marker	(Text)	PM
k	hour in day (1 – 24)	(Number)	24
K	hour in am/pm (0 – 11)	(Number)	0
z	time zone	(Text)	Pacific Standard Time
'	escape for text	(Delimiter)	
''	single quote	(Literal)	'

The count of pattern letters determines the format.

Presentation	Usage
(Text)	If the count of pattern letters is 4 or more, use the full form. If the count is less than 4 pattern letters, use short or abbreviated form if one exists.

Presentation	Usage
(Number)	Represents the minimum number of digits. Shorter numbers are zero-padded to this amount. Year is handled specially; that is, if the count of y is 2, the year will be truncated to 2 digits.
(Text & Number)	M is the only format field that can be number or text. If the count of M is 1 or 2 it is to be read as a number. If the count of M is 3 it is read as an abbreviated month name. If the count of M is 4 it shows the full month name.

Any characters in the pattern that are not in the ranges of ['a'..'z'] and ['A'..'Z'] will be treated as quoted text. For instance, characters like ':', ',', ' ', '#', and '@' will appear in the resulting time text even though they are not embraced within single quotes.

Date Format	Result
yyyy.MM.dd G 'at' hh:mm:ss z	1996.07.10 AD at 15:08:56 PDT
EEE, MMM dd, 'yy	Wed, Jul 10, '96
h:mm a	7:08 PM
hh 'o' 'clock' a, zzzz	07 o' clock PM, Pacific Daylight Time
K:mm a, z	7:00 PM, PST
yyyy.MMMM.dd GGG hh:mm aaa	1996.July.10 AD 12:08 PM

Automatic Activity – E-mail BIC



The E-Mail BIC provides the capability to generate e-mails from an EPX workflow by connecting to an E-mail server.

In a new Enable installation (with the Services Framework installed), the Scheduled Imports, Scheduled Exports, and Email Notification Process workflows include the Email BIC activity. Any additional workflow should use the Send Email Callout BIC to send e-mails. The reason for this is because there are a lot of settings required in the Email BIC that would have to be configured in each workflow that uses it. The Send Email Callout BIC simply requires the basic information (e.g., From Address, To Address, Subject, Body, Attachments, etc.) and does not require any Email-server specific configuration since it routes the e-mail requests through the Email Notification Process Workflow.

Configuring the E-mail BIC

To properly configure the E-mail BIC and optimize it for your particular needs, you must edit the EmailBic.config file installed on the BIC Manager host system in <EPX>\bin\agents\ <hostname>.

To configure the E-mail BIC to either use the SSL or TLS for a secure inbound or outbound transaction, refer to “Configuring SSL/TLS” section.

The EmailBic.config file has an SMTP default ‘from’ e-mail address used for outgoing e-mails. The unit of the inbound mail-polling interval is in seconds.

The code sample below displays the default EmailBic.config file contents:

```
# Available threads
minThreads=1
maxThreads=100
# The smtp default 'from' e-mail address, which is used for
# outgoing e-mails.
sys.smtp_from_email_address=name@domain
# The inbound mail polling interval in seconds
sys.inbound_mail_poll_period=60
# Specifies the number of times the BIC should retry to
connect
# if the InboundStore failed due to an
AuthenticationFailedException.
# set the value to 0 if no retries are to be attempted
sys.retry.count=15
# Specifies an underscore to dot notation conversion or
vice-versa
# "sys.workitem.property.delimiter=." - will do underscore
to dot conversion
# "sys.workitem.property.delimiter=" - no conversion will
be done for null or empty space value
sys.workitem.property.delimiter=.
# contains the server certificate chain
sys.truststore.file=
sys.truststore.password=changeit
# contains the client certificate chain
sys.keystore.file=
sys.keystore.password=changeit
# comma separated list of SSL providers
ssl.provider.list=com.sun.net.ssl.internal.ssl.Provider
internationalization.encodingname=ISO-8859-1
```

Important: The key names should never be modified.

Thread Configuration

The minThreads and maxThreads parameters are used to control the number of concurrent threads. As threads are created, they are added to a pool. The minThreads and maxThreads settings determine the size of this pool.

- minThreads is the minimum number of instances of the BIC that will be created. The lower the minThreads value, the more time will be needed for the BIC to start a new instance. By increasing the minThreads value, memory usage is increased but processing time is decreased when a BIC instance is required. A value of zero (0) means that the thread pool can become empty when no work is done. The valid values are from 0 to 32767, with 0 as the default value.
- maxThreads is the maximum number of instances of the BIC that will be created. By increasing the maxThreads value, the maximum number of concurrent BIC instances is increased, denoting that a higher volume of work can be passed to this BIC. Increasing the number too high could cause the BIC to consume

the system resources on the server or servers hosting the BIC Manager and the instance of the EPX Application Server being used. The `maxThreads` should have a value of at least one (1), so that at least a thread can be created to process work items. The valid values are from 1 to 32767, with 32767 as the default value.

If `minThreads` is greater than `maxThreads`, their values will automatically be swapped and a warning will be displayed in `bic.log`.

Connection Retry Configuration

The value of `sys.retry.count` specifies the number of times the BIC should retry to connect if an inbound e-mail failed due to some authentication errors.

The value of `sys.retry.count` is set to 15 by default; hence, the system will attempt to connect for up to 15 times in case of inbound e-mail failure.

Setting the `sys.retry.count` value to 0 instructs the system not to retry to connect at all.

Default Outbound E-mail Return Address Configuration

When configuring an E-mail BIC activity to send e-mail, one of the parameters is the e-mail's From address. If no address is entered, the E-mail BIC uses the default outbound return address indicated in the `EmailBic.config` file. Before using the E-mail BIC to send e-mail, you should configure this address accordingly. Upon installation, the default address is `name@domain`. Change this address to the address you would like to use, and then remove the spaces before and after the equal sign, so that the key/value pair has the format:

```
sys.smtp_from_email_address=<your_address>@<your_domain>
```

Inbound Mail Polling Interval Configuration

When using an E-mail BIC activity as a starting point in a flow, by default it checks for new mail every 60 seconds. You can shorten or lengthen this polling interval by adjusting its value in the `EmailBic.config` file. To do this, change the value of the `sys.inbound_mail_poll_period` key to the desired number of seconds.

Underscore to Dot Conversion

Keys in the message body of inbound e-mails can make use of the conversion from underscore to dot notation.

By default, the `sys.workitem.property.delimiter` is equal to “.” to convert all occurrences of underscore in the work item property keys to dot.

Setting the `sys.workitem.property.delimiter` to null or empty space value results to no conversion.

Important: Do not forget to restart the web services after configuring the E-mail BIC.

Keystore and Truststore Configuration

The `EmailBic.config` contains the keystore and truststore files which contain the public and private keys for authentication and data integrity. Modify the location and password of the truststore and keystore files for the server or client certificate chain to reflect the appropriate settings specified after generating the keystore and truststore.

```
# contains the server certificate chain
sys.truststore.file=d:/Enterworks/EPX/bin/certs/server.keystore
sys.truststore.password=changeit
```

```
# contains the client certificate chain
sys.keystore.file=d:/Enterworks/EPX/bin/certs/server.keystore
sys.keystore.password=changeit
```

Note: The keystore and truststore directory and password values specified above are examples only. You can change this to reflect the values used by your EPX installation.

Refer to the “Configuring SSL/TLS” section for information on creating and verifying the keystore and truststore files, as well as setting up the config.properties or the EmailBic.config file to enable a secure data setting for SSL or TLS.

Configuring SSL/TLS

The E-mail BIC user can enable a secure E-mail connection by setting up E-mail BIC to use either SSL or TLS for a secure inbound or outbound transaction.

SSL is the common protocol for managing a secure message transmission over the Internet. SSL utilizes a cryptographic system that uses two keys to encrypt data - a public and a private key.

TLS has two layers. One is the TLS Record Protocol which provides connection security with or without an encryption method, and the TLS Handshake Protocol which allows the server and the client to authenticate each other and negotiate encryption and cryptographic keys before data is exchanged between them.

Refer to the procedure below on how to enable the security settings for the inbound and outbound security settings.

Creating a Keystore File

This section describes how to generate the keystore file along with the truststore file using the keytool utility to allow the following: self-authentication, data integrity, and authentication services.

The keystore can manage two types of entries -- the Key Entry and the Trusted Certificate Entry. The Key Entry holds sensitive cryptographic key information in a secure, protected format; while the Trusted Certificate Entry contains a single public key certificate and can be used to authenticate other parties.

The truststore file contains the public keys which are stored as signer certificates used for authentication and data integrity. This file verifies and authenticates the legitimacy of the client operating on behalf of the user.

1. Create a keystore file importing the matching certificate from the mail server. You can use the keytool utility located in your <EPX>/jre/bin folder.
2. Open the keytool utility by clicking on keytool.exe. Execute the following command:

```
keytool -import -trustcacerts -alias <alias> -file
<cert_file> -keystore <keystore_file>
```

This command generates both the keystore and the truststore files.

Note: Keep in mind that the <alias> is the name you want to specify for the certificate when you import it into the keystore file. Since your keystore file may contain more than one certificate, the alias can be used to identify each individual certificate in the keystore file.

The <cert_file> is the name of your .cer file (certificate file matching the mail server's)

The <keystore_file> is the name of your .keystore file

Verifying the Keystore File

The keytool utility can also be used to verify if the certificate was successfully imported into the keystore file. To use this tool, perform the following command:

```
keytool -list -keystore <keystore_file>
```

Verify if the alias you specified during the import is included in the output. Refer to the output example below:

```
<alias>,<Date>,<type of entry>  
Certificate fingerprint (MD5):XX:XX:XX:XX:
```

Configuring the config.properties and/or the EmailBic.config Files

Open the config.properties or the EmailBic.config file and add the appropriate values for the keystore file path and keystore password keys under the server and client's certificate chain settings.

For a system-wide configuration which includes the E-mail Notification and E-mail BIC, add the keystore file path in the directory below:

```
<EPX>/bin/config.properties
```

To configure your settings only for the E-mail BIC, add the keystore file path in the directory below:

```
<EPX>/bin/agents/<machine_name>/EmailBic.config
```

Enabling SSL/TLS for Inbound and Outbound

After having configured and generated the files needed to ensure a secure data transmission either by SSL or TLS, you can set the security settings either for inbound or outbound by doing the following:

- On the Inbound tab, select the SSL Enabled checkbox to set the security settings for the inbound transaction to SSL;
- On the Outbound tab, select SSL in the Security options to set the security settings for the outbound transaction to SSL. Alternatively, you may choose to use instead the TLS as your secure transaction option by selecting TLS.

TLS and SSL are not interoperable. However, a message sent with TLS can be handled by a client that handles SSL but not TLS.

When using the MS Exchange Server as the mail server which is configured to use the secure SMTP, the outbound security option for the E-mail BIC should be set to TLS and not SSL.

Enabling TLS for the outbound secure setting works appropriately when using the MS Exchange Server 5.5 version.

Sending E-mail with the E-mail BIC

When an E-mail BIC activity is not the starting point in a flow, it is used for generating and sending e-mail messages. The BIC activity can generate an e-mail message using the work item property keys and values it receives from the previous activity in the process flow. An e-mail sent by the BIC can include attachments such as files that were appended to work items and files existing in the file system.

The E-mail BIC uses existing e-mail accounts for sending messages. In order to send messages generated by an E-mail BIC activity, an e-mail account must be available to the activity.

The E-mail BIC requires values for the following fields: the mail **Server**, the **Userid**, **Password**, the message **Body**, and the **Result Set**. In addition, the **Outbound** tab requires **SMTP From** which should contain a valid e-mail address and the **SMTP Port Number** field which contains the port number being used by the SMTP. The **Recipient** address and e-mail **Subject** should also be provided.

E-mail Formatting Options

On the **Outbound** tab of the E-mail BIC editor (discussed in the “Outbound Tab” section), the **To**, **Cc**, **Bcc**, **Subject**, and **Body** boxes can contain constant values, work item property table keys, or both. The use of work item property table keys allows the work item in the flow to supply the appropriate e-mail address, and allows customization of the subject line and body text. Work item property table keys must be fully qualified and enclosed within percent signs (%) to be valid.

For example, you are the Order Fulfillment manager for a software retailer called K&L Software Supply and you are using an E-mail BIC activity in a flow used for order processing, and you wish to send an order confirmation e-mail to each customer after their order is received. Assume also that the work item property table includes the following keys for the customer’s first name, last name, e-mail address, and order number, respectively:

```
customer.firstname
customer.lastname
customer.email.address
order.no
```

In the **To** box, you would type:

```
%customer.email.address%
```

In the **Subject** box, you might type:

```
Order Confirmation for %customer.firstname%
%customer.lastname% - Order # %order.no%
```

In addition to work item property table keys, the body text can also contain HTML content if desired. <HTML> and </HTML> opening and closing tags are not required. In the **Body** box, you might type:

```
To: %customer.firstname% %customer.lastname%
Thank you for your order. We have begun processing the
order and we will notify you when it is shipped.
Your order number is %order.no%. Please refer to this
number in any correspondence regarding your order.
<EM>K&L Software Supply</EM>
```

When the outgoing message is generated, the keys resolve to the corresponding values from the work item.

Note: MIME types are applied to the sent HTML messages as follows: text/html is used if defined on the mail server; otherwise, text/plain for HTML may be used. If this is not successful, plain text (text/plain) e-mail is sent.

Receiving E-mail with the E-mail BIC

When used as a starting point in a process flow, an E-mail BIC activity creates a new work item for every incoming e-mail message received. The E-mail BIC activity retrieves inbound e-mail from an accessible mailbox on a POP3 or IMAP mail server that you designate, converts the e-mail into a work item, and initiates the process flow. The E-mail BIC polls mailboxes for incoming mail based on the polling frequency defined in the EmailBic.config file. At the default setting, this polling occurs once every sixty seconds. (For instructions on changing the polling interval, see “Inbound Mail Polling Interval Configuration,” section on page 4.)

In addition to converting an e-mail message to a work item, an E-mail BIC activity can also append any file attachments received with the message to the work item that it creates. An E-mail BIC activity can also save a copy of the e-mail message body and any attachments to an external file system.

There are three ways to use the E-mail BIC to receive messages. One, you can create a separate mailbox for each process flow that starts with an E-mail BIC activity. Two, you can configure a single E-mail BIC activity that uses a decision point activity (DPA) to invoke the appropriate subflow based on the content of the e-mail message. Three, you can also have multiple process flows start with an inbound BIC that has the same server settings.

Each process flow will have a work item created for each e-mail sent to the specified e-mail account. Each process flow should then screen the work items to ensure that only the appropriate ones are processed. The difference of this approach to the second one (DPA to route work item) is that each process screens e-mail work items and only accepts the ones it needs, rather than just route the work item to the appropriate subflow. Additional processes that use the same e-mail account can be created without any modification to the existing flows. A new name for each work item is also possible.

Dynamic Creation of Work Item Property Table Keys

Upon retrieving a new message, the E-mail BIC automatically creates a new work item, with the work item property table keys for the To, From, Cc, Bcc, Subject, and Body fields of the message. In addition, content in the body of the message can be used to create additional work item property table keys. Body content can be formatted in either of the following ways:

- Each key/value pair on a separate line. For example:

```
lastname=Sardinia
firstname=Diane
birth=01/01/1980
address=Main+Street
city=Arlington
state=VA
```


EnterWorks Workflow Activities and Callout BIC Framework

- As a string, with each key/value pair delimited by an ampersand (&), and replacing spaces with the plus sign (+), for example:

```
lastname=Sardinia&firstname=Diane  
birth=01/01/1980&address=Main+Street&city=Arlington&state=V  
A
```

When a message is received in the mailbox specified on the **Inbound** tab in the E-mail BIC editor (discussed in “Inbound Tab” section on page 17), the E-mail BIC retrieves the message, parses the message body, and generates a work item with a work item property table containing:

- Key/value pairs, which record the **To**, **From**, **Cc**, **Bcc**, **Subject**, and **Body** values from the incoming message
- Valid key/value pairs for this work item type, derived from the body of the incoming message

In the work item property table, all key names are prefixed by the result set name entered in the **Result Set** box on the **Inbound** tab. For example, for an E-mail BIC activity with a result set name of **emailresults**, the work item property table for the resulting work item would contain keys like the following:

```
emailresults.to=datapool@some-company.com  
emailresults.from=joe.campbell@some-company.com  
emailresults.cc  
emailresults.bcc  
emailresults.subject=Personal Information  
emailresults.body=lastname=Sardiniafirstname=Dianebirth=01/  
01/80  
address=Maine streetcity=Arlingtonstate=VA  
emailresults.content.lastname=Sardinia  
emailresults.content.firstname=Diane  
emailresults.content.birth=01/01/1980  
emailresults.content.address=Main Street  
emailresults.content.city=Arlington  
emailresults.content.state=VA
```

You can refer to any of these keys in subsequent flow activities.

Inbound Tab

The **Inbound** tab contains the **POP3**, **IMAP**, and **SSL Enabled** options, the **Server**, **Port**, **Userid**, **Result Set**, **Directory**, **E-mail Content** fields, the **Password** button, and the **Include e-mail attachments to the work item**, **Save e-mail content and attachments to the file system**, and **Set user specified keys only** checkboxes.

Note: The **Inbound** tab can only be enabled when the **Starting Point** checkbox is selected under the **Details** tab.

General | Details | Expiration | **Inbound** | Outbound | Custom Action

☒ POP3 ☐ IMAP ☐ SSL Enabled

Server:

Port:

Userid:

Result Set:

Directory:

E-mail content:

☒ Include e-mail attachments with the work item

☒ Save e-mail content and attachments to the file system

☐ Set user specified keys only

To configure the E-mail BIC activity to retrieve new e-mail messages from the mail server and use the e-mails to generate work items:

1. Specify the correct protocol for the mailbox from which the E-mail BIC activity will retrieve messages by clicking the corresponding radio buttons of available options; i.e., POP3, MAP.
2. Click the SSL Enabled checkbox if you want to allow a more secure connection for data transaction between client and server.

Selecting this option entails performing an added procedure. For more information on configuring the Inbound setting to SSL, refer to “Configuring SSL/TLS” section.

3. In the Server field, type the name of the mail server where the mailbox is located. For example, pop.some-company.com.
4. In the Port field, type the port number that the inbound setting will use.
5. In the Userid field, type the user name for the mailbox. For example, datapool.
6. In the Result Set field, type a name to be used as prefix for the key names (created from the incoming e-mail) in the generated work item’s work item property table. For example, if you type “emailresults” in the Result Set box, the work item property table for the work item generated from the incoming e-mail would contain key/value pairs like the following:

```
emailresults.to=datapool@some-company.com
```

```
emailresults.from=joe.campbell@some-company.com
emailresults.cc
emailresults.bcc
emailresults.subject=Personal Information
emailresults.body=lastname=Sardiniafirstname=Dianebirth=01/
01/ 80address=Maine streetcity=Arlingtonstate=VA
emailresults.content.lastname=Sardinia
emailresults.content.firstname=Diane
emailresults.content.birth=01/01/1980
emailresults.content.address=Main street
emailresults.content.city=Arlington
emailresults.content.state=VA
```

In this example, the first six work item property table entries are automatically generated from the To, From, Cc, Bcc, Subject, and Body fields of the incoming mail. The last six work item property table entries are created from the formatted key/value pairs in the body of the message. Subsequent activities can operate on any of these keys and values.

7. Click the **Password** button. A box containing the E-mail Account Password and Confirm Password fields appear.
8. Enter the e-mail account password and confirm the password.
9. Click **OK**.
10. To include e-mail attachments (if any) with the generated work item, select the Include e-mail attachments to the work item checkbox.
11. To save the content of each e-mail to the file system, along with any attachments, select the Save e-mail content and attachments to the file system checkbox.

If you set the Save e-mail content and attachments to the file system option enabled, you must fill out the following fields:

- **Directory** – Specify where the E-mail BIC activity should save the e-mail and any attachments by entering the directory path in the provided field.
 - **E-mail content** – Specify the filename under which the contents of the e-mail message should be saved. EPX supports the creation of unique file names. If a unique filename is not used for saving the e-mail contents, then each time the process flow is started, the previous file will be overwritten. To have EPX create a unique filename for each saved e-mail, type the string “%sys.workitem.version.uid%” (without the quotes).
12. Click the Set user specified keys only checkbox to store the content of the inbound e-mail message body under keys as defined in the e-mail.
 13. Save the data entered by clicking **Apply**. Clicking **OK** will also save the data entered and exit the Email BIC editor. To cancel saving the data entered, click **Cancel**.

Outbound Tab

The Outbound tab contains the Wait for reply checkbox; Security and E-mail Format options; the SMTP Server, SMTP Port Number, Requires authentication; SMTP Password button; SMTP From, To, Cc, Bcc, Subject,

and Body fields; the SMTP Userid, Include work item file attachments to the e-mail, and Include files as attachments to the e-mail checkboxes; and the Files table.

The screenshot shows the 'Outbound' configuration tab for an email activity. It includes fields for SMTP settings, email format, authentication, and recipient information. The 'Body' field contains a template for an order confirmation email. There are checkboxes for including work item file attachments and other files as attachments, and a table for listing specific files to attach.

To configure the setting for outgoing e-mail:

1. Click the **Wait for Reply** checkbox if you want to hold sending the work item to the next activity until a reply to the outgoing e-mail is received.
2. Specify the email security setting for the Outbound BIC by clicking the corresponding radio button of the **Security** option. If you select **SSL**, you will perform an added procedure. For more information on configuring the Outbound setting to **SSL**, refer to “Configuring SSL/TLS” section.
3. Specify the e-mail message format by clicking the corresponding radio button of an **Email Format** option (e.g., **Rich Text (HTML)** or **Plain Text**).
4. In the **SMTP Server** field, type the name of the mail server used for sending e-mail. For example, `mail.some-company.com`.
5. In the **SMTP Port Number** field, enter the port number of the SMTP server. The default port number is 25.
6. Click the **Requires authentication** checkbox to enable authentication of user and password login for outbound E-mail.

7. In the SMTP Userid field, specify the user ID for the mailbox that will be used for sending messages.
8. Click the **SMTP Password** button. A dialog box containing Password and Confirm password fields appears.
9. On the dialog box, enter and confirm the user ID password.
10. Click **OK**.
11. In the SMTP From field, specify the sender address that will appear in the From field for all e-mail messages sent by this E-mail BIC activity. The address specified in this field will be the default address where the reply e-mails will be sent to. For example, orders@some-company.com.

Note: If you do not specify an address, the default address contained in the EmailBic.config file is used. See “Configuring the E-mail BIC,” on page 2 for more information on the EmailBic.config file.

12. In the To, Cc, and Bcc fields, designate the recipient address or addresses for the outgoing messages. Multiple addresses can be entered in one box if they are separated by commas.
13. Specify the subject and body of the e-mail in their corresponding fields.
14. To include work item attachment with or files as attachment to the e-mail, click the corresponding checkboxes below the Body box.

Note: If you select the Include files as attachment to the e-mail option, the Files table is enabled. Specify the files to be used as attachment by clicking the **Browse** button and then **Add Row**.

15. Save the data entered by clicking **Apply**. Clicking **OK** will also save the data entered and exit the Email BIC editor. To cancel saving the data entered, click **Cancel**.

Automatic Activity – Epim Enable Handler BIC



The Epim Enable Handler BIC is a difficult to configure activity for interfacing with Enable and is not recommended for use.

Automatic Activity – EPIM Manage Item BIC



The Epim Enable Handler BIC provides a generic interface to Enable repositories to allow adding, updating or deleting records in an Enable repository. The update and delete operations require specifying the search criteria which could potentially affect a large number of records. Newer Callout BIC activities provide the majority of functionality this BIC provides but in a more isolated manner resulting in a lower risk.

Automatic Activity – EPIM Manage Process BIC



The EPIM Manage Process BIC provides the ability to launch background processes in Enable and then check their progress later in the workflow. This BIC is mostly out-of-date and therefore should not be used.

Automatic Activity – File Attachment BIC



The File Attachment BIC provides the ability to attach files to a work item or to populate work item properties with details of files that may have been attached by a previous activity, such as an FTP GET operation or inbound Email-BIC configured to allow attachments.

In general, workflows should not be developed that have file attachments since they would mostly be accessible within work item viewers which are no longer the common way to interact with a work item in an EPX workflow. Instead, the user interacts with the anchor repository record and if there are attachments, they will likely be in the form of digital assets linked to that record. If files need to be attached to e-mails, the Send Email Callout BIC has a provision for including file attachments using the fully-qualified file path.

The configuration file named:

<drive>:\Enterworks\EPX\bin\agents\<agentServer>\FileAttachmentBic.config specifies defaults for the following properties:

- **directoryPath** – default directory path. If the Directory Path field is not defined in a File Attachment BIC activity, this default value will be used.
- **fileNames** – default file names. If the File Names field is not defined in a File Attachment BIC activity, these default values will be used.
- **resultsKey** – default results key. If the Results Key field is not defined in a File Attachment BIC Activity, this default value will be used.
- **debugEnabled** – enables debug output to the bic.log if set to 'true'

An example file is shown below:

```
# FileAttachmentBic.config file

# Default directory path
directoryPath=d:/tmp

# Default list of file names
fileNames=*.sql,*.txt

# Default results key
resultsKey=FileAttachmentResults

# Enable debug reporting
debugEnabled=true
```

To add the File Attachment activity to a process flow, perform the following steps:

1. Select the Automatic Activity icon and click on the desired screen location. The BIC Selection dialog appears.
2. Select 'File Attachment' and click OK.
3. Open the property editor for the activity.
4. Set the name of the activity on the General tab.
5. Click the Settings tab.

6. Optionally enter a directory path if all files to be imported are located on the same disk. If files from multiple disks are to be attached within a single activity, the Directory Path field must be left blank and the names in the File Names field must be fully qualified. The directory path can be a literal string or a work item property reference (surrounded by '%'), or a combination of the two. Since the backslash character may be used to designate a Windows directory, the escape character has been set to '^'. Wildcard characters ('*') cannot be specified in the directory path.
7. Enter the names of the files to be attached in the File Names field. Separate file names with carriage-returns, or commas, or tabs. Spaces cannot be used as a delimiter since directory and file names can contain spaces. File names can include literal values, work item property references (surrounded by '%') and wildcard characters ('*') or any combination of the three. The wildcard character behaves the same as it does in DOS or UNIX list directory commands.

Since the backslash character may be used to designate a Windows directory, the escape character has been set to '^'. The file names can include directory paths but the directory paths cannot contain wildcard characters ('*').

8. Optionally specify a work item property field name in the Results Key field under which the attachment processing results will be stored. The results key can contain a literal value, work item property references (surrounded by '%'), or a combination of both.

The File Attachment BIC will generate the following properties:

- **<resultsKey>.numberOfFiles** – number of files found and attempted to be attached
- **<resultsKey>.<file#>.file** – path and name of file attempted to be attached
- **<resultsKey>.<file#>.status** – status of attach operation:
 - ❑ **Attached** - file was successfully attached
 - ❑ **Not a file** – the file is a directory or some other non-regular file
 - ❑ **Unable to read file** – the file could not be read
 - ❑ **Unable to copy file** – the file could not be copied to the attachments directory
 - ❑ **Exception occurred during attachment copy: <message>** - an unexpected error occurred during the copying of the file to the attachments directory.

where:

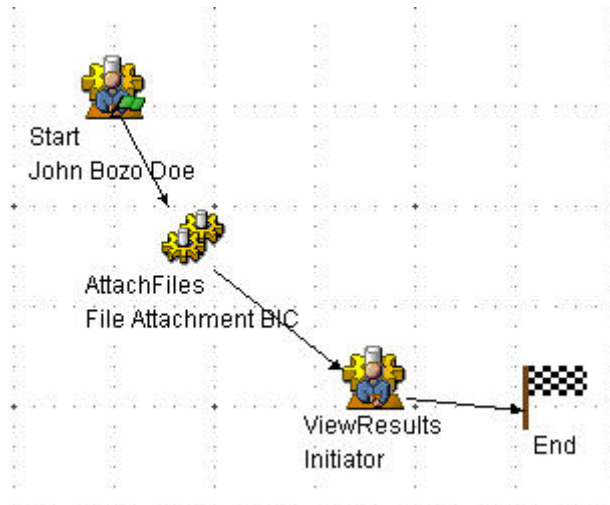
- **<resultsKey>** is the optional work item property key specified in the Results Key field of the property editor. If the key is left blank, these properties will be saved under the individual names.
- **<file#>** is the file number, starting with zero, to be imported. For example, if the Results Key is set to 'Attachments' and three files are specified for attachment, the entries may look like:

```
Attachments.numberOfFiles=3
Attachments.0.file=myFirstFile
Attachments.0.status=Attached
Attachments.1.file=dummyFile
Attachments.1.status=Unable to read file
Attachments.2.file=myThirdFile
Attachments.2.status=Attached'
```

If the Results Key is not set, the same entries would appear as:

```
numberOfFiles=3
0.file=myFirstFile
0.status=Attached
1.file=dummyFile
1.status=Unable to read file
2.file=myThirdFile
2.status=Attached'.
```

9. Click OK to save the activity.
10. Connect the activity to the other activities in the flow. The File Attachment BIC cannot be made a starting point within a flow. For example:



11. Save the process flow.

Automatic Activity – Flow Correlation BIC



The Flow Correlation BIC allows the user to create a Master Flow and a Detail Flow. A Master Flow is a process flow that has the ability to create one or more work items within a Detail Flow and then wait for those Detail Flows to complete. A Detail Flow is a process flow that can be initiated by a Master Flow and will send a signal to the Master Flow when a work item reaches the end of the flow.

The Flow Correlation BIC can be configured to:

- Send a Request - create an outgoing request to generate a new work item in another flow
- Wait for a Request - handle incoming requests to generate new work item(s) in the flow
- Send a Reply - send a response to a request, indicating that the Detail Flow has completed execution
- Wait for a Reply - wait for a response from a Detail Flow to which a request was sent

The Flow Correlation BIC facilitates creating complex workflows that are difficult to implement, troubleshoot and maintain. Therefore, it is recommended that this BIC not be used in new workflow development.

Automatic Activity – FTP BIC



The FTP BIC provides the capability to send or receive files to or from an FTP server.

There are two flavors of this BIC, with the new one including support for SFTP servers. In the current installations, both BICs are provided and by default the old version (with no SFTP support) is active. Some manual installation/configuration steps must be followed in order to make use of the new version.

Settings tab:

1. In the Remote Directory field, type the path of the directory the activity should access on the remote host. For example:

/private/hr/data/payroll/fy2001

Note: For **Get** request, the remote directory is the location of the file or files that will be retrieved. The remote destination directory for the file transfer is used for **Put** request. An error occurs if the specified remote directory does not exist.

In cases where a static directory path is not sufficient, use work item hash table keys enclosed in percent signs (%) to specify part or all of the remote directory path. This is not applicable to a activity not set as a starting point.

For example, if a company's human resources department maintains individual directories for each employee, with the directories named using the format "Lastname_Firstname"; and if the work item hash table contains

employee name keys called `employee_first` and `employee_last`; then the path typed in the **Remote Directory** box must be:

`/private/hr/employees/%employee_last%_%employee_first%`

Note: Work item hash table keys cannot be used with **Get** requests if the FTP BIC activity is a starting point in the process flow.

2. In the **Remote File** field, specify the filename for the file(s) to be retrieved. This step is optional for **Get** action or when the activity is set to send files.

If a new filename is not specified, the original will be used. The filename entered can contain the following types of name elements:

- Standard legal filename characters. For example, if you type “holidayparty.doc”, the activity will attempt to retrieve or send the file holidayparty.doc.
- One or more asterisks (*) used as wildcards. For example, if you type “benefits_*”, the activity will attempt to retrieve or send all files beginning with the string “benefits_”. When a wildcard is used to retrieve files its behavior changes if used as a starting point or as non-starting point. If used as a starting point, a new work item will be started for each file that shows up. If used as a non-starting point, all of the files will be transferred and the results for all transfers will go into the single work item.
- For non-starting point activities, enclosed one or more work item hash table keys in percent signs (%). For example, if you type “2001_%employeeNo%.xls” and the work item has a hash table key called `employeeNo` with a value of “210540”, the activity will attempt to retrieve or send the file 2001_210540.xls.

Important: When executing a **Put** request with an FTP BIC activity, the remote file will be overwritten if it has the same name as the file being sent.

3. In the **Local Directory** field, type the path of the directory the activity should access on the local host. You can also click the **Browse...** button and select the desired directory.

Note: If the activity will perform a **Put** request, the local directory is the location of the file or files that will be sent. The local destination directory for the file transfer is used for **Get** request. An error occurs if the specified local directory does not exist.

When the BIC is used in a clustered environment, make sure that the folders and/or files that will be indicated are existing in and pointing to where the BIC Manager server is installed. For non-starting point activities, work item hash table keys enclosed in percent signs (%) can be used to specify part or all of the local directory name, if necessary.

4. If the FTP BIC activity will be used to send files, type the name of the file to be sent in the **Local File** box. If instead the activity will retrieve files, then this step is optional and is only necessary if files need to be saved locally under different filenames than they had remotely. If a new filename is not specified, then the original filename is preserved. For a **Get** request, make sure that local file and the file being retrieved must have different filenames to avoid the local file from being overwritten. The filename entered can contain the following types of name elements:

- Standard legal filename characters.
- One or more asterisks (*) used as wildcards. When a wildcard is used to retrieve files its behavior changes if used as a starting point or a non-starting point. If used as a starting point, a new work item

will be started for each file that shows up. If used as a non-starting point, all of the files will be transferred and the results for all transfers will go into the single work item.

- For non-starting point activities, one or more work item hash table keys, enclosed in percent signs (%).
5. From the **Transfer Type** dropdown list, select the type of the file to be transferred (i.e., ASCII, Binary).
 6. In the **LIST Gaps** field, specify the number of blank gaps. This step is optional for Put request.

Note: Blank gaps are spaces between groups of characters when viewed in a directory listing on the remote FTP server. The **Get** operation will fail if the number entered in the **LIST Gaps** field is incorrect. A bic.log file will contain an error message identifying the file name can be retrieved .

7. The number of gaps can be determined by manually establishing an FTP connection to the FTP server and typing the command: dir <filename> where <filename> is the fully qualified file of interest. Then, count the number of blank areas between characters. For example, if the remote file were a purchase order form called po.doc located in the FTP directory, then the FTP command to be typed is:

```
dir ftp/po.doc
```

Supposing that the output of the command appeared as follows:

```
-rw-r--r-- 1 vdbeps eps 2490217 Dec 11 10:41 ftp/po.doc
```

For this example, the number of **LIST gaps** must be set to 8.

If the directory listing on the remote FTP host contains more or fewer data fields than the example shown above, the number of gaps must be adjusted accordingly. For the example above, if the number in the **LIST Gaps** box entered was 6, then the file name will appear as “11 10:41 ftp/po.doc” in the error message. This indicates that the **LIST Gaps** value must be increased by 2, making the correct value 8.

8. In the **Result Set** field, type a result set name for the FTP BIC activity.

When a file is successfully transferred, the transfer is logged as a key/value pair in the work item hash table, in the format:

```
<Result_Set>_File<n>=<filename>
```

Where, <Result_Set> is the result set name entered in the **Result Set** box and File <n> identifies the position in the transfer order for the file. <n> is an incrementing number starting with 1. <filename> is the name of the file transferred, as it was saved in the destination file system. (That is, if the original filename has been changed due to a destination filename being specified, then it is the new destination filename that appears as the value.)

For example, if an FTP BIC activity has a result set name of `getPayroll`, and the activity successfully transfers two files saved as “fy2001_210540.xls” and “fy2001_212031.xls”, then the following key/value pairs are added to the work item hash table:

`getPayroll_File1=fy2001_210540.xls getPayroll_File2=fy2001_212031.xls`

9. By default, the option **Generate error if 0 files match the criteria** is selected to prompt the user with an error message when no files are found in the remote directory to match the search criteria in the **Remote File** text box. This scenario happens in a **Get FTP** action when a wildcard is used to search for a file in an empty directory or when the standard filename entry is non-existent in the remote directory.

Note: If this option is not selected, no error messages will be generated upon execution.

10. To attach a copy of each transferred file to the work item, select the **Attach to work item** checkbox.
11. Click the **Delete file from server** checkbox to delete all retrieved files from the remote FTP server. This step is only applicable to **Get** requests. Selecting this option when executing a **Put** request will not delete sent files from the local file system.
12. Click the **Include Subdirectories** checkbox to send eligible files from multiple directories located beneath the selected local directory. This step is only applicable to **Put** requests.

Note: When sending files, the FTP BIC activity will search all directories contained within the directory you entered in the **Local Directory** box and will send any files that match the filename you entered in the **Local File** field.

13. Save the data entered and proceed to another tab by clicking **Apply**. Clicking **OK** will also save the data entered and exit the FTP BIC editor. To cancel saving the data entered, click **Cancel**.

Automatic Activity – HTTP BIC



The HTTP BIC provides the capability to send HTTP request to a web server and capture the response in a work item property.

Settings Tab

For new HTTP BIC activities, the default **Settings** tab contains the **Download** and **Upload** options; the **Transfer Type** list; and the **Remote Host**, **Local Directory**, **Local File**, and **Interval** fields.

The screenshot shows a dialog box with five tabs: General, Details, Expiration, Settings (selected), and Custom Action. The Settings tab contains the following fields and controls:

- Action:** Two radio buttons, 'Download' (selected) and 'Upload'.
- Remote Host:** A text input field.
- User ID:** A text input field, with a 'Password' button to its right.
- Transfer Type:** A dropdown menu currently showing 'ASCII'.
- Local Directory:** A text input field, with a 'Search' button to its right.
- Local File:** A text input field.
- Interval:** A text input field followed by the text 'seconds (Starting Points Only)'. This field is disabled when the 'Upload' action is selected.

At the bottom of the dialog are four buttons: OK, Cancel, Apply, and Help.

The Download option is selected by default. When the Upload option is selected, the Settings tab also contains the Remote Directory and Remote File fields, while the Interval field is disabled.

1. Select the action you want to implement by clicking the corresponding radio button.

Important: If the activity will be used as a starting point in the process flow, then you must select the Download option. HTTP BIC starting point activities can only be used to download files. You can select either of the two options for non-starting point HTTP BIC activities.

2. In the Remote Host field, type the URL address to which the HTTP BIC activity should connect. Note that the URL address depends on how the activity will be used. If the HTTP BIC activity will be used to upload files, type the address of either the CGI script or the Java servlet that the Web server uses for accepting file uploads. For example: <http://www.some-company.com/servlet/uploadServlet>.
3. If the activity will be used to download files, type the URL address of the file to be downloaded. There are two ways in which to specify download addresses with the HTTP BIC:
 - If the HTTP BIC activity will download a file with a fixed filename in a fixed location, type the complete URL filepath for that file. For example:

<http://www.some-company.com/products/dailyupdate.html>
 - If the HTTP BIC activity will not be used as a starting point, you can also use keys from the work item hash table to define part or all of the URL filepath. Keys must be enclosed in percent signs (%) in order to be recognized by the BIC. For example, to download a file specified in the value of a work item hash table key called filename, you might type:

<http://www.some-company.com/downloads/%filename%>

Likewise, if the entire URL filepath were specified in a work item hash table key called `url_filepath`, you would enter in the **Remote Host** field the following:

`%url_filepath%`

Note: Rather than specifying the remote Web server by typing its domain name, the server has a static IP address, which you can use instead (e.g., `http://123.123.123.123`) followed by any additional filepath text. Doing this prevents any DNS-related errors from inhibiting the execution of the HTTP BIC activity.

The **Description** field on the **General** tab can be used for noting the domain name of the IP address entered. You can connect to a secure server by typing an HTTPS URL in the **Remote Host** field, provided that the security settings have been configured for the EPX installation that contains the BIC Manager of the HTTP BIC being used. Security configuration is normally performed during installation, but if this was not done, you can manually configure security by following the instructions outlined in the Administration guide.

4. If user authentication is needed, type in a valid user name in the in the **User ID** field. The specified user name is used to connect to the remote Web Server.
5. Click **Password**. Type in the necessary information in the **Password** and **Confirmation** fields. Note that the remote Web Server will first verify the given user id and password. If the information supplied is valid, connection will push through. The values in the **User ID** and **Password** fields will not matter if user authentication is not needed.

Important: Contact the Systems Administrator of the remote Web Server to obtain a valid user ID and password.

6. Select the type of file to be transferred from the **Transfer Type** dropdown list (e.g., ASCII or Binary).
7. If the HTTP BIC activity will be used to upload files, you must specify the remote directory for the upload. Providing a new name for the transferred file is optional.

- In the **Remote Directory** field, type the directory path where uploaded files will be saved on the remote Web server. Each directory in the path should have the format “/**<directory>**”. For example:

`/uploads/product_news`

In cases where using a static directory path is not sufficient, work item hash table keys enclosed in percent signs (%) can be used to specify part or all of the remote directory path (provided that the HTTP BIC activity is not a starting point).

- If files transferred by the HTTP BIC activity must be saved on the remote Web server under a filename different from the local filename, then type the new filename in the **Remote File** field. If the **Remote File** field is left empty, then the local filename will be used by default when saving files remotely. If the HTTP BIC activity is not a starting point, work item hash table keys enclosed in percent signs (%) can be used to specify part or all of the filename.
8. In the **Local Directory** field, type the path of the directory that the HTTP BIC activity should access on the local host or click **Browse...** and select the directory. If the activity will be used to upload files, the local directory is the location of the file or files that will be uploaded. For download file activity, the local directory is the destination directory for the file download.

Note: Make sure that the local directory you have specified does not exist to prevent encountering an error during the execution of the HTTP BIC activity. When using this BIC in a clustered environment, make sure

the folders and/or files that will be indicated are existing in and pointing to where the BIC Manager server is installed. As with the **Remote Directory** field, for non-starting point activities, work item hash table keys enclosed in percent signs (%) can be used to specify part or all of the local directory name, if necessary.

9. In the **Local File** field, specify the name of the file to be uploaded. This step is optional for download file action and is only necessary if files need to be saved locally under different filenames than they had remotely. If a new filename is not specified, the original filename is preserved. As with the **Remote File** field, the filename entered can contain the following types of name elements:
 - Standard legal filename characters.
 - For non-starting point activities, one or more work item hash table keys, enclosed in percent signs (%).
10. If the HTTP BIC activity is a starting point in the process flow, type the polling interval in the **Interval** box. The polling interval is the frequency, measured in seconds, with which the HTTP BIC activity will attempt to download the file specified in the **Remote Host** field.
11. Save the data entered and proceed to another tab by clicking **Apply**. Clicking **OK** will also save the data entered but will exit the HTTP BIC editor. To cancel saving the data entered, click **Cancel**.

Automatic Activity – JDBC BIC



The JDBC BIC provides the capability to execute a SQL query or stored procedure in EPX or EPIM database. The Java Database Connectivity (JDBC) BIC provides the ability to execute a single SQL statement or a stored procedure against a JDBC-compliant database (or against an Open Database Connectivity (ODBC)-compliant database, if a native JDBC driver exists for it). The JDBC BIC can also be used to connect to the Content Exchange Server RDBMS.

To configure a JDBC BIC activity correctly, you must know the name of the JDBC driver you are using, the connection string, and the user login ID and password for the JDBC-compliant database. You must also know the SQL statement that you wish to use or the name of the stored procedure in the database and all of its parameters (if any). Finally, you are required to enter a name for the hash table where the results will be placed.

The JDBC BIC does not provide much control over what work item properties are created so it should be used sparingly. If the need is to retrieve data from Enable and/or to invoke a stored procedure (which should at a minimum return success/fail status), the SQL Set Properties BIC should be used instead.

Settings Tab

The default Settings tab contains the Driver, Connection String, User ID, Password, and Database Encoding fields; the Access Type options (SQL and Stored Procedure); and the SQL Statement and Result Set fields.

If you select the Stored Procedure option, the SQL Statement field disappears and the SP Name and Parameters fields appear, along with the DataType, IO Type, and Precision lists, and the Value field.

1. In the **Driver** field, type the name of the JDBC driver that the activity will use. Remember that in order to successfully use this driver, its .jar file must be copied to the proper directory, and its

classpath must be referenced in the JDBCbic.config file (see “Classpath Configuration,” on page 4 for more information).

2. Type the proper connection string for the driver in the Connection String field.

For SQL Server, the connection string can take the following format:

```
jdbc:jtds:sqlserver://<db_hostname>:<port>/<database_name>;SelectMethod=cursor
```

In this format, <db_hostname> is the hostname of the server, which is set to localhost by default. The <port> refers to the number the server is listening on. It defaults to the SQL Server standard port number (1433, if the default was accepted during installation). Finally, <database_name> refers to the name of the database which defaults to EPX (if the default was accepted during installation).

3. In the User ID and Password fields, type the username and password required in order to connect to the database.
4. If you have enabled double-byte character support (DBCS) for EPX, the Database Encoding field is editable, allowing you to change the encoding format if necessary. The default value is 8859_1.

If you need to connect to a JDBC-compliant database that does not share this same encoding format, you must replace the default with the correct format.

For non-DBCS EPX installations, the Database Encoding field appears dimmed and the default encoding format is not editable.

5. Select the desired Access Type option, either SQL or Stored Procedure, and proceed to appropriate section, either “Configuring the SQL Option,” below or “Configuring the Stored Procedure Option,” on page 12.
6. Save the data entered and proceed to another tab by clicking Apply. Clicking OK will also save the data entered and exit the JDBC BIC editor. To cancel saving the data entered, click Cancel.

Configuring the SQL Option

If you select SQL as the Access Type option, do the following steps:

1. In the SQL field, type the necessary SQL statement. JDBC BIC

General | Details | Expiration | **Settings** | Custom Action

Driver: sun.jdbc.odbc.JdbcOdbcDriver

Connection String: jdbc:odbc:flow

User ID: flow

Password: ****

Database Encoding: 8859_1

Access Type: ☒ SQL ☐ Stored Procedure

SQL Statement:

```
select * from flowtest where type='one'
```

Result Set: sr Max Rows:

OK Cancel Apply Help

Note: If you want to use the percent sign in the SQL statement, escape it by putting a backslash in front of each percent sign; for example,

Select * from P_ACTIVITY_PROPERTY where PROPERTY_KEY like '\%Password\%'

resolves to

Select * from P_ACTIVITY_PROPERTY where PROPERTY_KEY like '%Password%'.

- In the Result Set field, type the fully qualified name for the result set. This is the location in the hash table where the results will be placed (e.g., <user>.<directoryName>).

To pull any piece of information from a hash table, enter the fully qualified hash table entry name.

Note: To specify a percent sign in the name itself, escape it by putting a backslash in front of it, for example: Pct\% resolves to Pct%

To specify a backslash in the name itself, escape it by putting another backslash in front of it, for example, `\\Results` resolves to `\Results`.

3. The Max Rows field specifies the maximum number of rows to be retrieved into the result set in the work item. You can enter a numeric value or a work item property key.
 - For numeric values, if you specify a value that exceeds the number of rows, then it will return all the rows for that query.

Type "0" to return zero rows or "-1" to return all rows.

If the input is invalid, the Max Rows field adapts the default value specified in the `JDBCbic.config` file. Invalid values include negative numbers other than "-1", mixed decimals and non-numeric input.

In the `JDBC.config` file, the Max Rows default value of 100 is configurable.

- For work item property keys, type in the property name enclosed with the "%" sign. For example, if you will use the work item property name `Personal_Age` type in `%Personal_Age%` in the Max Rows field. The query will return the number of rows specified in the `Personal_Age` field in the viewer. If the work item property specified in the Max Rows field does not exist, the query will result to an error.

Configuring the Stored Procedure Option

If you select Stored Procedure at the Access Type option, do the following steps:

1. In the SP Name field, enter the name of the stored procedure the JDBC BIC activity will access in the database that you entered.

General | Details | Expiration | **Settings** | Custom Action

Driver: sun.jdbc.odbc.JdbcOdbcDriver

Connection String: jdbc:odbc:flow

User ID: flow

Password: ****

Database Encoding: 8859_1

Access Type: ☐ SQL ☒ Stored Procedure

SP Name: GetFlowTest

Parameters

1) VarChar (In)= one	Up
	Down
	Remove
	Update

DataType: IO Type: Precision: Value: Add

Result Set: sr Max Rows:

OK Cancel Apply Help

Note: If the name of the stored procedure contains spaces, enclose the name in double quotes. For example, "Sales by Year"

- Enter the data that the stored procedure will use by selecting the data type from the DataType dropdown list. The data type must correspond to the first value expected by the stored procedure.
- In the IO Type list, select the input/output types. The following are the available input/output types:
 - In describes the parameter to the stored procedure as input.
 - Out describes the parameter to the stored procedure as output.
 - InOut describes the parameter to the stored procedure as both input and output.
- If you select either Decimal or Numeric as the data type, you can specify the precision for the value if desired by clicking the appropriate number in the Precision list.
- In the Value field, enter the value for the parameter that you have just configured. You can enter a static value or you can enter a variable that will be defined in the work item. To define

the value with a variable, you must escape the variable name with percent signs (%). For example, if the work item contains a variable named param.value, it must be entered as %param.value%.

6. To enter data with null values, type in NULL (all uppercase letters) or do not enter any value at all. On the other hand, string "null" can be specified by typing in the null string with at least one letter in lowercase, e.g., null, NULL, nULL or Null.

Note: Entering of a null value allows the JDBC BIC to create work item properties for result columns that are null. The created workitem property contains an empty string.

7. When you have finished configuring a parameter, click Add to add the parameter to the Parameters field.
8. Repeat the steps 1 to 7 to add each of the parameters expected by the designated stored procedure.

You can rearrange the parameters that you have added to the Parameters field by moving them up and down in the list. To move a parameter, click the parameter to select it and then click Up or Down to move the parameter up or down in the list, respectively.

To delete a parameter from the list, select the parameter to be deleted and then click Remove.

To update a parameter from the list, select the parameter to be updated. Then configure the values in the Data Type, IO Type, Precision, and Value fields. After configuring the values in the fields, click Update.

9. In the Result Set field, enter the fully qualified name for the result set. This is the location in the hash table where the results will be placed (e.g., <user>.<directoryName>).

To pull any piece of information from a hash table, enter the fully qualified hash table entry name.

Note: To specify a percent sign in the name itself, escape it by putting a backslash in front of it; for example, Pct\% resolves to Pct%.

To specify a backslash in the name itself, escape it by putting another backslash in front of it; for example, \\Results resolves to \Results.

10. The Max Rows field specifies the maximum number of rows to be retrieved into the result set in the work item. You can type in a numeric value or a work item property key.
 - For numeric values, all the rows for that query will be returned if you specify a value that exceeds the number of rows.

Type "0" to return zero rows or "-1" to return all rows. If the input is invalid, the Max Rows field adapts the default value specified in the JDBCbic.config file. Invalid values include negative numbers other than "-1", mixed decimals and non-numeric input.

In the JDBC.config file, the Max Rows default value of 100 is configurable.

- For work item property keys, enter the property name enclosed with the "%" sign. For example, if you will use the work item property name Personal_Age type in

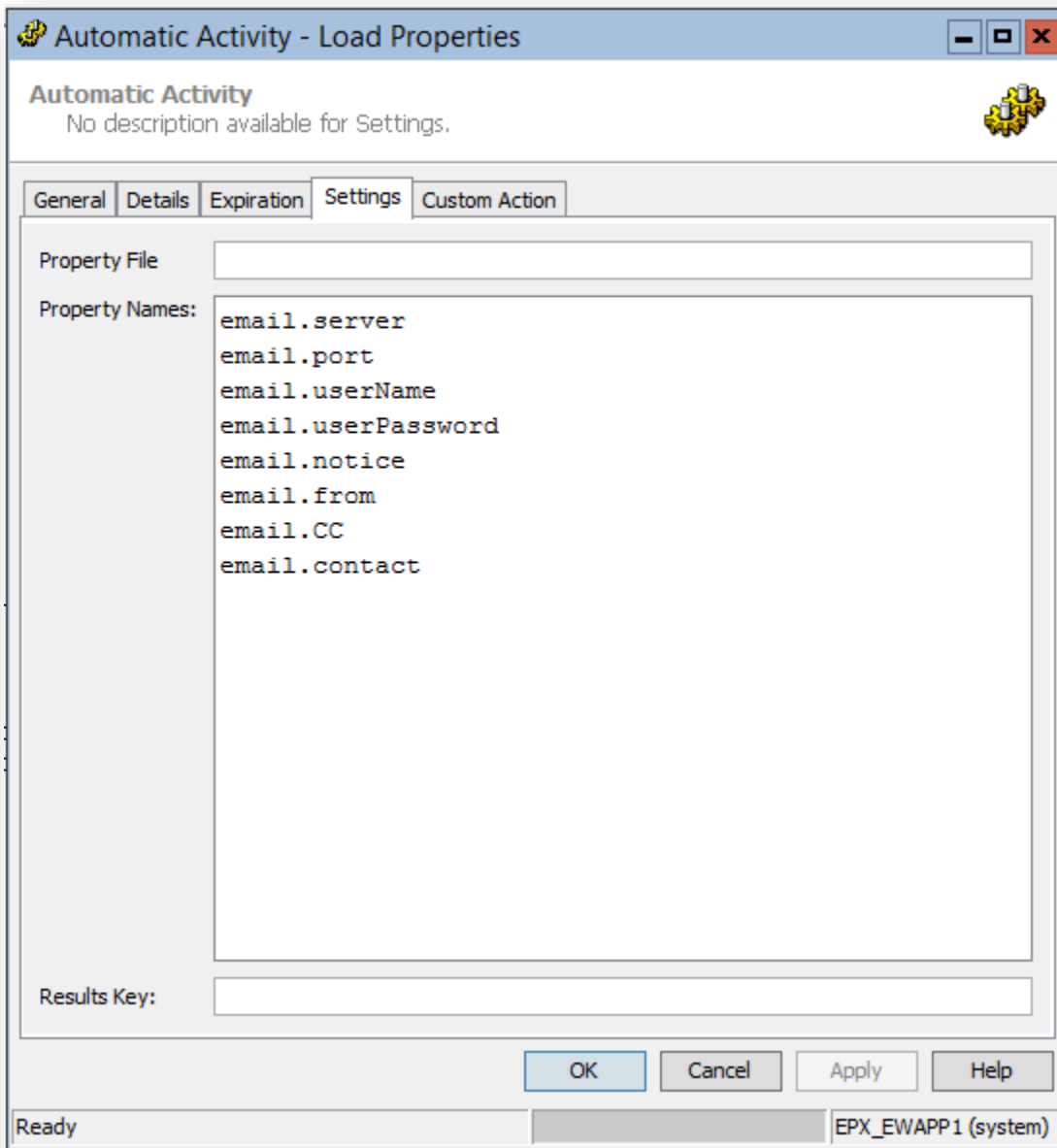
%Personal_Age% in the Max Rows field. The query will return the number of rows specified in the Personal_Age field in the viewer. If the work item property specified in the Max Rows field does not exist, the query will result to an error.

11. Save the data entered and proceed to another tab by clicking Apply. Clicking OK will also save the data entered and exit the JDBC BIC editor. To cancel saving the data entered, click Cancel.

Automatic Activity – Load Properties BIC



The Load Properties BIC populates work item properties for the designated properties in the Enterworks.properties file. The names of the work item properties will match the names of the properties in the file. Each property must be listed by name on the settings tab.



If the Property File and Results Key fields are empty, the defaults from the LoadPropertiesBIC.config file are used:

```
# Copyright (c) 2001-2015 Enterworks Acquisition, Inc. All
rights reserved. "Enterworks" and the "e." logo are
registered trademarks and "Enterworks Enable Process
Exchange" is a trademark of Enterworks Acquisition, Inc.
#

# configurable property file for the LoadPropertiesBIC
```

```
# Available threads
minThreads=1
maxThreads=100

#
# Properties file to be loaded
#

fileName=D:/Enterworks/EPX/bin/conf/Enterworks.properties
#propertyName
#resultsKey
#debugEnabled
```

If the resultsKey is not set, the work item properties created will exactly match the properties listed in the Property Names field.

Automatic Activity – Process Properties BIC



The Process Properties BIC provides the means to define or update work item properties. The UI for the Settings tab is difficult to manage, plus there are size limitations with what can be set. Therefore, use of this BIC should be avoided. Instead, the Set Properties Callout BIC should be used instead. If the setting of the work item properties is dependent on any data in Enable, the SQL Set Properties Callout BIC should be used.

Automatic Activity – Purge Completed Workitems BIC



The Purge Completed Workitems BIC provides the means for a workflow to remove the completed work items after a designated period of time. Without building this purge into the workflow or requiring an administrator to periodically purge completed work items from EPX, the growth of the completed work items will eventually have a significant negative impact on workflow performance. It is strongly recommended that each workflow include the Purge Completed Workitems BIC activity. If the volume of work items is low (e.g., no more than one per hour or one per day), the purge operation can be placed in-line within the main flow. For higher work item volumes, the Purge activity should be connected to a Scheduler BIC starting point that is configured to run no more frequent than once per hour. The purge operation is a resource-intensive operation so it should not be run more frequently than prescribed above.

Settings Tab

The Settings tab is used to specify the completed work items to be included in the Purge Completed Work Items activity as well as the login information of the user authorized to perform the purge.

1. The Purge Completed Work Items BIC deletes all work items completed within the time range that you specify in Completed Expiration Time. The time range can be specified in Days, Hours, or Minutes.

Note: You can set all Completed Expiration Time values to zero, this prompts the Purge Completed Work Items BIC to delete all completed work items in the database when the activity is run.

2. Specify the login information of the user authorized to perform the purge activity in the Purge user name and Purge user password fields.

Note: Both Purge user name and Purge user password are required. Only the “system” user has the privileged access to use the Purge Completed Work Items BIC.

Automatic Activity – Scheduler BIC



The Scheduler BIC provides the ability to create new workitems in a workflow on a periodic basis (e.g., every 10 minutes, once per hour, once per day, etc.) or to delay the processing of a work item for some period of time when monitoring for completion of an external operation or until a specific date and time.

Using the Scheduler BIC as a Starting Point

If the Scheduler BIC is used as the starting point in a process flow, the Settings tab will include both a Begin Date and Time field, and a Repeating Enabled checkbox.

1. If you select the Repeating Enabled checkbox, you can establish a schedule for the BIC to repeatedly initiate a process flow and select a date to stop the BIC from initiating a process flow.

The screenshot shows the 'Settings' tab of a dialog box. It contains the following fields and controls:

- Begin Date & Time:** 04 - 29 - 2003, 4 : 51 PM
- Repeating:**
 - ☒ Enable
 - Frequency: By Minute (dropdown)
 - By Minute section: Every: 1 minute(s)
- End Date & Time:** ☒ 04 - 29 - 2003, 4 : 51 PM
- Execution Dates:**
 - Last Scheduled Execution: Never
 - Next Scheduled Execution: Apr-29-2003 4:51 PM
 - Reset button
- ☒ Dispatch if expired
- Buttons: OK, Cancel, Apply, Help

2. You can select the Dispatch if expired checkbox to permit the work item to be sent to the next activity in the process flow, in the event that the BIC manager is unavailable to process the work item at the scheduled time. The work item will be sent as soon as the BIC manager is able to do so.

Using the Scheduler BIC in the Body of a Process Flow

The Scheduler BIC can dispatch work items on dates specified at runtime or at design time. Design-time dates are specified within the activity editor in Design Console. Runtime dates are specified within a work item property contained in work items arriving at the activity.

Static Date

To configure a static date (i.e., a date specified at Design-time) that will apply to all work items arriving at the activity, take the following steps:

1. Select the Dispatch Date radio button.
2. Select the Specific Date radio button to enable the date and time fields
3. Specify the dispatch date in the date and time fields.

The screenshot shows a settings dialog box with tabs: General, Details, Expiration, Settings, and Custom Action. The 'Settings' tab is selected. Under the 'Dispatch' section, the 'Dispatch Date' radio button is selected. Below it, the 'Specific Date' radio button is also selected, with date and time fields set to '04 - 29 - 2003' and '4 : 51 PM'. The 'Dynamic Date' radio button is unselected. The 'Work Item Key' field is empty, and the 'Date Format' is set to 'MM-dd-yyyy HH:mm'. The 'Format Test' shows '04-29-2003 16:51'. The 'Delay' section is also visible, with the 'Delay Interval' radio button selected, and fields for Week(s), Day(s), Hour(s), Minute(s), and Second(s) all set to 0. The 'Delay until the next' radio button is unselected, and the 'Delay until the next workday at' radio button is also unselected.

Dynamic Date

To configure a dynamic date (i.e., a dispatch date that is specified at run-time) that will cause the dispatch date to be extracted from work items that arrive at the activity, do the following steps:

1. Select the Dispatch Date radio button.
2. Select the Dynamic Date radio button to enable the Work Item Key and Date Format fields.

Note: The Dispatch Date feature of the Scheduler BIC works similarly as the Delayed Send feature of all automatic activities. However, they have a difference in purpose. If both of these features are used in an activity, the Scheduler BIC settings will determine when a specific event will occur while the Delayed Send feature delays the sending of a work item by the specified amount of time once the work item is ready to be sent to the next activity.

3. Specify the Work Item Key that will contain the dispatch date in arriving work items.

The screenshot shows a dialog box with the following sections and controls:

- Dispatch Section:**
 - ☒ **Dispatch Date**
 - ☒ **Specific Date:** 04 - 29 - 2003, 4 : 51 PM
 - ☐ **Dynamic Date**
 - Work Item Key:** (empty text field)
 - Date Format:** MM-dd-yyyy HH:mm (dropdown menu)
 - Format Test:** 04-29-2003 16:51
- Delay Section:**
 - ☐ **Delay Interval**
 - ☒ **Delay**: Week(s) 0, Day(s) 0, Hour(s) 0, Minute(s) 0, Second(s) 0
 - ☐ **Delay until the next**: Sunday at 4 : 51 PM
 - ☐ **Delay until the next workday at**: 4 : 51 PM

Buttons at the bottom: OK, Cancel, Apply, Help.

4. Select or modify the format of the date by either choosing one from the dropdown list or entering an alternative (the value in the combo box is editable). The Date Format is where the user will select the format of the date that is stored in the key specified in the Work Item Key field. In this example, we will use the default format MM dd yyyy HH:mm.

The Format Test field contains the current date and/or time formatted using the format selected in the Date Format field. Each time a new format is selected, the value is updated to reflect the newly selected format.

5. If you do not wish to specify a dispatch date, you may opt to use the Delay option. This option allows you to move a work item to the next activity in the process flow after a specified time has passed since receipt at the Scheduler BIC.

General Details Expiration Settings Custom Action

Dispatch

☐ Dispatch Date

☐ Specific Date: 04 - 29 - 2003 5 : 00 PM

☒ Dynamic Date

Work Item Key: UserData.InputDate

Date Format: MM-dd-yyyy HH:mm

Format Test: 04-29-2003 17:00

Delay

☒ Delay Interval

☒ Delay

Week(s)	Day(s)	Hour(s)	Minute(s)	Second(s)
0	0	0	0	0

☐ Delay until the next

Sunday at 5 : 00 PM

☐ Delay until the next workday at

5 : 00 PM

OK Cancel Apply Help

- Click Delay and enter a value in at least one of the fields to delay the work item from being move to the next activity in the process flow for the time period specified.
- Click Delay until the next to delay the work item from being moved to the next activity in a process flow until the day and time specified. Use the drop-down menus to select the day and time.
- Click Delay until the next workday at to delay the work item from being moved to the next activity in a process flow until the specified time on the next workday.

Automatic Activity – SOAP Request BIC



The Simple Object Access Protocol (SOAP) BIC provides a mechanism for process flows to make use of this protocol in order to send or retrieve information from a remote source. SOAP defines the use of Extensible Markup Language (XML) and Hypertext Transfer Protocol (HTTP) to access services, objects, and servers in a platform-independent manner. Moreover, the SOAP BIC provides client access to existing SOAP services.

This BIC has been made obsolete by newer protocols.

Settings Tab

The Settings tab contains the Endpoint URL, Method Namespace URI, URI Encoding Style, Method Name, Action URI, Result Key, Number of Parameters text fields, and the Parameters list.

Each field can be substituted with work item properties by surrounding the property name with percent signs. In this case, its parameters will be referred to as dynamic parameters.

Note: The percent sign is used to identify work item properties. In order for the work item properties to be included in the statement, it should be enclosed with percent signs. So if you want to use the percent sign in the SQL statement, put a backslash in front of each percent sign, for example: `\%productName\%` resolves to `%productName%`.

1. In the URL text field, type in the Endpoint URL that identifies the SOAP service. For example, `http://ww6.borland.com/webservices/BorlandBabel/BorlandBabel.exe/soap/IBorlandBabel`.
2. In the URI field, type in the SOAP URI of the namespace for the service. For example, `urn:BorlandBabelIntf-IBorlandBabel`.
3. Type the name of method to be invoked from the SOAP server in the Method Name field. For instance, `BabelFish`.
4. The Action URI is an optional input for SOAP service. If necessary, the user may type it in the Action URI text field. For example, `urn:BorlandBabelIntf-IBorlandBabel#BabelFish`.
5. In the Result Key field, type in the base name of work item where all results are to be stored.
6. When the parameters are defined dynamically, the Number of Parameters field for service should be set. Otherwise, if the parameters are fully defined in the parameters section, this field should be left blank.
7. In the Parameters list, each parameter is defined on a separate line if the number of parameters is fixed. The name, value, and type of each parameter are also specified.

If the number of parameters is dynamic, only the first parameter should be defined and must serve as the template for each parameter. The asterisk indicates where the parameter index (starting with 0) should be inserted. For example, the template `SOAP.Parm*.key` indicates that the parameter names will be defined as `SOAP.Parm0.key`, `SOAP.Parm1.key`, `SOAP.Parm2.key`, and so on, in the work item.

Note: The following are the supported data types for SOAP BIC: string, java.lang.String, boolean, value, double, integer, int, long, short, decimal, base64Binary, hexBinary, byte, value, dateTime, and QName.

8. To add an entry to the list of parameters, click the Add button and then type in the name, value, and type of the parameter to be added.
9. To remove an entry from the list of parameters, select the parameter to be removed and then click the Remove button.

Automatic Activity – Telnet BIC



Telnet is the standard Internet protocol for logging into a host computer from a remote location. It runs on top of TCP/IP and allows a computer to emulate a terminal during the remote login session.

Provided that the EPX host system is running a Telnet daemon, the Telnet BIC initiates Telnet sessions and executes commands that you specify in a Telnet BIC activity.

This BIC has been mostly rendered obsolete by newer protocols.

Prompt Tab

The Prompt tab contains the Userid Prompt, Password Prompt, and Command Prompt fields.

The screenshot shows a configuration dialog box with six tabs: General, Details, Expiration, Prompt, Settings, and Custom Action. The 'Prompt' tab is selected. It contains three text input fields: 'Userid Prompt:' with the value 'login:', 'Password Prompt:' with the value 'Password:', and 'Command Prompt:' with the value '\$'. At the bottom of the dialog are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

1. In the Userid Prompt field, type the user ID prompt as it appears when connecting to the remote system.
2. Likewise, in the Password Prompt field, type the password prompt as it appears when connecting to the remote system.
3. In the Command Prompt field, type the character that is used as the command prompt on the remote system.
4. Click the Settings tab.

Settings Tab

The Settings tab contains the Host, Port, User ID, Password, Command, Result Set, and Timeout fields.

The screenshot shows a configuration window for a Telnet BIC activity. It features a tabbed interface with the following tabs: General, Details, Expiration, Prompt, Settings, and Custom Action. The 'General' tab is currently active. Below the tabs are several input fields: 'Host' (empty), 'Port' (containing '23'), 'User ID' (empty), 'Password' (empty), 'Command' (empty), 'Result Set' (empty), and 'Timeout' (containing '60'). At the bottom of the window are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

1. In the Host field, type the hostname of the remote system that this Telnet BIC activity should connect to.
2. In the rather unlikely chance that the remote host uses a port other than 23 for Telnet connectivity, delete the default port number and type the correct number in the Port field.
3. In the User ID field, type a valid username for logging in to the remote system and then type the corresponding password in the Password field.
4. In the Command field, type the command or commands that this Telnet BIC activity should execute on the remote system. To execute multiple commands, separate the commands with semicolons (;).
5. In the Result Set field, type the fully qualified name of the hash table key for the Telnet BIC activity's result set. For example:

user.directoryName

The result set is a string that stores the output of the Telnet session. You would see this same output on-screen if you were manually interacting with the Telnet session. The result set name and its output string are a key/value pair. The result set string can be viewed by adding its key to a work item viewer.

Important: Result set names should not include spaces, or characters other than A-Z, a-z, 0-9, and the underscore character (_).

The exit codes from the semicolon-delimited commands are not written to the result set. The same is true for stdout (standard output) and stderr (standard error).

6. If needed, in the Timeout field, type the number of seconds that should be allowed for Telnet commands to complete. If you leave this field blank or enter a zero, then Telnet commands will not time out.

Automatic Activity – Universal Groovy BIC



The Universal Groovy BIC provides the ability execute Groovy scripts from EPX. This BIC should not be used. In general, scripting from within any of the Universal BICs should be avoided and instead either existing configurable Callout BICs should be used or a new Callout BIC should be created. If there is ever a need to use a scripting BIC then only the Universal Java BIC should be used.

Automatic Activity – Universal Java BIC

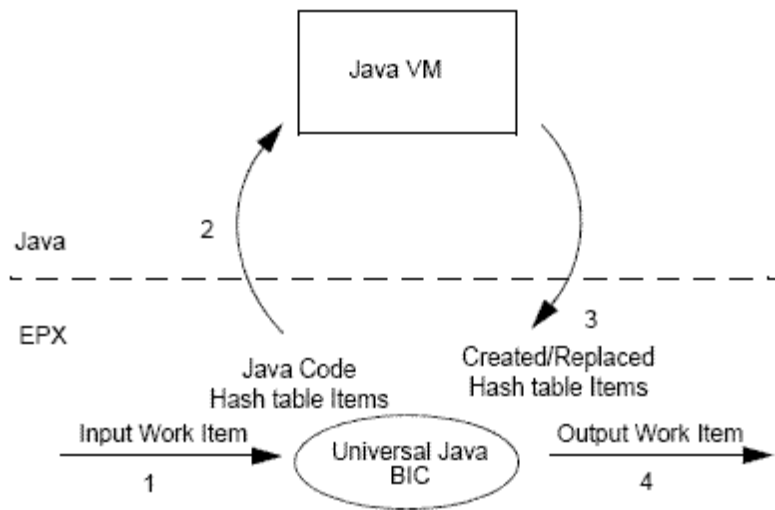


The Universal Java BIC provides the ability to execute any Java code within the EPX environment. In general, the use of this BIC should be one of last resort. Instead, existing configurable Callout BICs should be used if possible or new Callout BICs should be implemented to expand the pallet of available BICs or to ensure consistency in workflow development and troubleshooting if the operations are “one-offs”.

The Universal Java BIC uses the BeanShell interpreter, which is a Java source interpreter with object scripting language features. In addition to some scripting commands, BeanShell executes standard Java statements. For more information about the BeanShell Java source interpreter, please see

<http://www.beanshell.org>

The Universal Java BIC receives an input work item and executes code that creates or replaces data in the hash table. When the BIC has completed its transformation of the hash table data, the Universal Java BIC passes the modified work item on to the next activity in the process flow.



Automatic Activity – Universal Javascript BIC



The Universal Javascript BIC provides the ability execute Javascript scripts from EPX. This BIC should not be used. In general, scripting from within any of the Universal BICs should be avoided and instead either existing configurable Callout BICs should be used or a new Callout BIC should be created. If there is ever a need to use a scripting BIC then only the Universal Java BIC should be used.

Settings Tab

The Settings tab is used to define the Java code to be executed and what gets updated in the work item.

1. Enter the name of the property in the EPX hash table that you want to create or replace. You must know the name of the key. In the example below, the key User.fullName is used to create a new hash table key. The Code text box allows you to type Java code to transform information that is contained in the process flow.

General | Details | Expiration | Java Settings | Custom Action

Key: User.fullName

Code: `return User_firstName + " " + User_middleName + " " + User_lastName;`

OK Cancel Apply Help

Note: The key can be blank if you intend to return multiple hash table values. Instead of returning a string object, a hash table is returned. Each object in the hash table will have a work item property created where the name of the property is the Java BIC key plus the name of the hash table entry. If the Java BIC key is empty, the work item property name is just the name of the hash table entry.

The properties in the work item are made available as variables that can be used in the Java code that is written. The names of the variables are the same as the property name with dots replaced with underscores. In Java, the dot has specific meaning so it cannot be part of the variable's name. The underscore, however, will be used to reference work item properties in the Java code. For example, the property User.fullName would be referenced as User_fullName in the code.

Note: You must not have properties that are identical in name except for dots or underscores as doing so will make references to the variable in the Java code ambiguous.

In the **Code** text box, you can enter the standard Java statements and expressions. Statements and expressions are all of the things that you code inside a Java method, for example, variable declarations and assignments, method calls, loops, conditionals, math expressions, etc. You can declare and use methods in the Universal Java BIC just as you would in a Java class. These methods may also have dynamic (loose)

argument and return types. In the Universal Java BIC, you have the option of working with “loosely typed” variables. That is, you need not declare the types of variables that you use (both primitives and objects). The Universal Java BIC will still give you an error if you attempt to misuse the actual type of the variable.

Note: The Java code must return either a String (using toString() method for non-String objects) or a Hashtable if multiple properties are being created or updated.

2. The code used above uses first, middle, and last name keys and values and returns a new key and value to the hash table, for example, User.fullName. This example shows that you need to specify a hash table key, wherein the result is placed; and some code that returns a String, which is the result. For the sample code above, the hash table already contains these values: the Key contains User.firstName, User.middleName, and User.lastName and the Value contains Christopher, Michael, and Reigrut respectively.

If the user’s address needed to be set along with the full name, the key would be changed to User and the code would return a hash table containing the full name and address properties. Here is the code for this example with a Key User:

```
Hashtable ht= new Hashtable();
ht.put("fullName",User_firstName + " " + User_middleName +
" " +
User_lastName);
ht.put("address.street","123 Main St.");
ht.put("address.city","Main Town");
ht.put("address.state","Maine");
ht.put("address.zipcode","01234");
return ht;
```

This will result in the properties:

```
User.fullName = "Chris Michael Reigrut"
User.address.street = "123 Main St."
User.address.city = "Main Town"
User.address.state = "Maine"
User.address.zipcode = "01234"
```

If the key is left blank, the following properties would be created:

```
fullName = "Chris Michael Reigrut"
address.street = "123 Main St."
address.city = "Main Town"
address.state = "Maine"
address.zipcode = "01234"
```

3. The Java BIC has the capability to access, add, and delete attachments. To attach files to your Java BIC, click on the **Browse File** button and select the file. To add more attachments, click the **Add Row** button. Click the **Delete Row** button to delete the attachments.
4. Save the data entered and proceed to another tab by clicking **Apply**. Clicking **OK** will also save the data entered and exit the Universal Java BIC editor. To cancel saving the data entered, click **Cancel**.

Note: The best practice is to define a Hashtable object, populate it with any properties to be set in the work item, then return the object and keep the Key field blank.

Automatic Activity – Universal Pimql BIC



The Universal Pimql BIC provides the ability execute PIMQL scripts from EPX. This BIC should not be used. In general, scripting from within any of the Universal BICs should be avoided and instead either existing configurable Callout BICs should be used or a new Callout BIC should be created. If there is ever a need to use a scripting BIC then only the Universal Java BIC should be used.

Automatic Activity – Work Item Import and Export BIC



The Work Item Import and Export BIC can be used in many ways when creating a process flow or subflow. The BIC allows you to import information and append it to a work item hash table or export information from the hash table, in delimited, Extensible Markup Language (XML), and fixed width formats. The BIC can only act to import or export files in a single activity. You can create a process flow with multiple Work Item Import and Export BICs.

In general, the goal should be to minimize the creation of work item properties used by a workflow as the greater the number of properties, the slower the performance of the workflow. Therefore, this BIC should not be used.

Work Item Import/Export BIC Format Options

The Work Item Import and Export BIC offers great flexibility in configuring information for import to or export from a process flow. EPX allows you to use three formats -- Delimited, Fixed, and XML -- to organize information to be imported or exported. Each format has features and functions that can shape your decisions to use them in building a process flow.

Delimited

The delimited format allows you to select a pre-defined delimiter (comma, tab, semicolon, space, or other delimiter) to import into or export from EPX. If Other is selected as the delimiter, you must specify your own delimiter. All delimiters are found between the values being imported into a process flow or are placed between the work item properties in the export file to differentiate them from one another. You can also use double quotes around each imported value or exported work item property value in the event that one or more of the values contains the selected delimiter.

Import

You must also specify the document style when importing using the delimited format. This helps the BIC during the import process to determine how to parse a particular file for import. There are six different document styles supported:

- Table with keys
- Table without keys
- Array of values
- Key-Values, one line
- Key-Values, many lines
- Values only, many lines

You can specify the name (key) of the work item property that is created for each value in the import file, as well as the file keys that exist in the import file, depending on the document style selected. If no selections are provided in the Import Items region, all values found in the import file will be read and the corresponding work item property values created or updated. In this case, the import file keys will be used as the names (keys) of the work item properties.

Export

The layout of the export file contains either one, two, or several lines, depending on the delimiter specified and whether the option to include keys in the export is selected. If you select this option, the first line contains the export file keys (or all of the work item property keys if these are not specified). The second line contains the hash table values specified (or all of the work item property values if these are not specified). Each key or value is separated using the specified delimiter. If the option to include keys in the export is not selected, the first line contains the specified values (or all of the work item property values if these are not specified). An option to include system properties can also be selected, this will include the system properties in the export file.

XML

Specifying XML as the import or export format option allows you to process and possibly transform work item properties using industry standard XML and XSL notation. The Work Item Import and Export BIC reads XML files and converts the elements in the XML file into the dot "." notation that EPX uses for work item properties. Exporting work item properties into an XML file provides the reverse effect, work item properties using the dot notation is translated into the XML hierarchical representation. XSL and XPath can also be used to transform or search the XML file prior to it being processed by EPX. The syntax, notation, and functionalities of XML, XPath, and XSL are maintained by the World Wide Web Consortium (W3C). For more information on these, please visit <http://www.w3c.org>.

Import

When importing XML files into EPX, you have the option to use XPath, this allows you to search for a pattern in the XML file and import only a subset of the XML document based on the resulting match.

Work Item Import/Export BIC

Work Item Import/Export BIC Format Options 3

Below is an example of an XML import configuration at the Properties tab, using XPath to import the specific subset "BIO" of the sample XML document provided.

```
<?xml version="1.0"?>
<!DOCTYPE WorkItem [
  <!ELEMENT WorkItem (SYSTEM_ID,BIO,MyXMLVersion)*>
  <!ELEMENT SYSTEM_ID (#PCDATA)>
  <!ELEMENT BIO (AGE,INTEREST,SALARY)*>
  <!ELEMENT AGE (#PCDATA)>
  <!ELEMENT INTEREST (#PCDATA)>
  <!ELEMENT SALARY (#PCDATA)>
  <!ELEMENT MyXMLVersion (#PCDATA)>
  <!ATTLIST WorkItem processVersion CDATA #REQUIRED>
]>
<WorkItem processVersion="3.1">
<SYSTEM_ID>0-34661</SYSTEM_ID>
<BIO>
<AGE>young</AGE>
<INTEREST>lawn care</INTEREST>
<SALARY>above average</SALARY>
</BIO>
<MyXMLVersion>CONSTANT1</MyXMLVersion>
</WorkItem>
```

Export

An XSL file can be used to transform exported data from the default XML export format generated by EPX into another XML, HTML, or Text format. XML files can contain nested elements, as do work item properties in EPX. Using the same dot “.” notation that EPX uses for work item properties, you can force a nested XML structure in the exported XML. If no selections are provided in the Export Items text boxes, all work item property values are written to the export file with the work item property keys used as the XML tags. The nesting of work item properties is maintained as a nested XML structure for this case.

The default XML export format generated by EPX is wholly dependent on the content of the work item properties that exist at the activity. The default export format is dynamic in nature. By default, the export work item property keys have “WorkItem” as the root element. This root element tag name can be overridden by setting the Root element name in the BIC editor. A Document Type Definition (DTD) for the generated XML file can optionally be generated together with the XML file and can either be included internally with the XML file or saved as an external DTD file.

Fixed

Fixed format file export and import requires you to understand the structure of the information being manipulated extremely well. EPX does not have the ability to parse files to determine the data structure. You must know the column and line locations of keys and data to either export or import information from EPX.

Import

You can specify that an import file be parsed from specific line and column locations. You can import a fixed format file as a data table, where the line numbers for key value pairs would all be the same and the column positions would be different. If this option is not selected, the BIC will process the import file as if it were a semi-unstructured document (like an e-mail message) where the line numbers and column positions of different import values would be different. For the data table format, you have the option to ignore the first line of the import file if that line contains key names that you do not want to import.

The Work Item Key cell allows you to specify the name of the work item property to create or replace. The Start Line and Start Column allow you to specify where in the file to begin reading for a specific import value in the import file. The End Line and End Column allow you to specify where to stop reading for a specific import value in the import file.

Export

You have the option to specify that export file keys be included in the export file. If this option is selected, then the keys will be placed in the same starting column positions that are specified in the Export Items region of the tab.

The Export Items region for fixed format output must be provided. There is no way for EPX to determine how to write fixed format output for the participant for many reasons. The starting column must be specified for each file key specified.

The ImportExportAsynchBic.config File

The Work Item Import and Export BIC maintains information on default settings in a file named ImportExportAsynchBic.config. You can modify this file to change the default items for the BIC. You can find the ImportExportAsynchBic.config file in the <EPX>/bin/agents/<hostname> directory.

```
# user configurable property file for the Import/Export BIC
minThreads=1
maxThreads=100
# Import / Export specific properties
defaultDelimiter=,
defaultDirectory=../../bin/data
defaultXMLRootTag=WorkItem
```

defaultDelimiter

The default Delimiter for the Work Item Import and Export BIC is the comma (“,”)

character.defaultDirectory

The BIC will save exported files to the <EPX>/bin/data directory. The exact path will vary depending on the location to which you install EPX.

defaultXMLRootTag

This value establishes WorkItem as the default document type for XML files created by the Work Item Import and Export BIC.

Settings Tab

General Details Expiration Settings Export Import Custom Action

Direction: ☒ Export ☐ Import

File: Browse ...

OK Cancel Apply Help

1. Select the activity that the BIC will perform by clicking the corresponding **Direction** radio button (e.g., Import or Export).

If you select **Import**, the **Import** tab is enabled. See **Import Tab** section for more information.

If you select **Export**, the **Export** tab is enabled. See **Export Tab** section for more information. You also will be required to enter a string that specifies a unique file name. Each time the process flow is started, the previous file will be overwritten. EPX supports creating unique file names. You can use any replaceable parameter that returns a unique value; however, we suggest using this string:

`%sys.unique.file%`

This string enables EPX to create a unique file every time a flow using the Work Item Import and Export BIC is initiated. For example, you could create a file such as:

C:\TEMP\%sys.unique.file%.TXT

2. In the **File** text field, specify the location and name of the file you want to export or import. You can also click **Browse** and select from the **File** dialog box.
3. Save the data entered and proceed to another tab by clicking **Apply**. Clicking **OK** will also save the data entered but will exit the Work Item Import/Export BIC editor. To cancel saving the data entered, click **Cancel**.

Note: When this BIC is used in a clustered environment, make sure that the folders and/or files that will be indicated are existing in and pointing to where the BIC Manager server is installed.

Import Tab

Import allows you to configure the Work Item Import and Export BIC to specify the format of the import file. The available formats are delimited, XML, and fixed width. After you select a format, the BIC provides you access to only those options required to configure the export file.

Delimited

Use this option if you want to set the separator for values in files to be imported. The **Delimiter** dropdown list is enabled when you select the **Delimited** option.

General Details Expiration Settings Export **Import** Custom Action

Import Format

☒ Delimited ☐ XML ☐ Fixed

Delimiter: Comma

☐ Contains Quotes

Quote escape character: \

Document Style

Table with keys

DESCRIPTION,QUANTITY
Browser,1000

☐ Decode CRLF encoded as:

Import Items

Import File Key	Work Item Key

Add Remove

OK Cancel Apply Help

Delimiter

Select the field delimiters for files to be imported from the **Delimiter** dropdown list (e.g., **Comma**, **Tab**, **Semicolon**, **Space**, or **Other Delimiter**). If you select **Other Delimiter**, you must specify the delimiter in the text box to the right of the **Delimiter** dropdown list. If one or more of the work item property values contain the selected delimiter, enclose the values in double quotes.

Contains Quotes Option

Click the **Contains Quotes** checkbox if you want to enclose work item property values in quotation marks (“ ”). When you select this option, the **Quote Escape Character** field is enabled. Specify the symbol to escape double quotes. The default symbol is a backslash (\).

Document Style

The screenshot shows the 'Import Format' dialog box with the 'Import' tab selected. The 'Delimited' radio button is chosen. The 'Delimiter' is set to 'Comma'. The 'Contains Quotes' checkbox is unchecked. The 'Quote escape character' is set to '\'. The 'Document Style' dropdown menu is open, showing six options: 'Table with keys' (selected), 'Table without keys', 'Array of values', 'Key-Values, one line', 'Key-Values, many lines', and 'Values only, many lines'. A preview window to the right of the dropdown shows a sample of the selected style: 'DESCRIPTION,QUANTITY' and 'Browser,1000'. Below the preview, there are two columns labeled 'File Key' and 'Work Item Key'. At the bottom of the dialog are 'Add' and 'Remove' buttons. The bottom of the window has 'OK', 'Cancel', 'Apply', and 'Help' buttons.

The Document Style determines how to parse a particular file for import. Select the document style to be used for the delimited format from the Document Style dropdown list. Selecting a specific document style from the dropdown list enables the text area to the right of the style selector and displays a sample of the document style.

There are six different document styles supported:

Document Style

Description

Table with keys

First line contains the keys for the table. Subsequent lines represent the rows in the table:

```
Item_no,Description,Quantity
P750,PrintCo 750 Inkjet Printer,1
P820,PrintCo 820 Inkjet Printer,7
```

After import, the values are identified by the key and row number:

```
Item_no1 = P750
Item_no2 = P820
Description1 = PrintCo 750 Inkjet
Printer
etc.
```

Table without keys

Each line represents a row in the table:

```
P750,PrintCo 750 Inkjet Printer,1
P820,PrintCo 820 Inkjet Printer,7
```

Keys must be defined for each column in the Import Items dialog box:

```
Item
Description
Qty
```

After import, the values are identified by the key (specified in the Import Items dialog box) and the row number:

```
Item1 = P750
Item2 = P820
Description1 = PrintCo 750 Inkjet
Printer
etc.
```

Array of values

The values can be on one or multiple lines:

```
P750,PrintCo 750 Inkjet Printer,1
P820,PrintCo 820 Inkjet Printer,7
```

At least one key must be defined in the Import Items dialog box. All values are loaded and assigned to each key defined:

```
Catalog
```

After import, the values are identified by the key and index number:

```
Catalog1 = P750
Catalog2 = PrintCo 750 Inkjet
Printer
Catalog3 = 1
Catalog4 = P820
etc.
```

EnterWorks Workflow Activities and Callout BIC Framework

Document Style	Description
Key-Values, one line	<p>Each value has its own keyword. Keyword-value pairs can be on one or more lines (i.e., line breaks are treated like the delimiter). Each keyword must be unique (only the first occurrence of a keyword will be retained):</p> <pre>Item_no1,P750,Description1,PrintCo 750 Inkjet Printer,Quantity1,1 Item_no2,P820,Description2,PrintCo 820 Inkjet Printer,Quantity2,7</pre> <p>After import, the values are identified by the corresponding keywords:</p> <pre>Item_no1 = P750 Quantity2 = 7 etc.</pre>
Key-Values, many lines	<p>Each line has a keyword and one or more values.</p> <pre>Item_no,P750,P820 Description,PrintCo 750 Inkjet Printer,PrintCo 820 Inkjet Printer Quantity,1,7</pre> <p>After import, the values are identified by the corresponding keywords. If multiple values are specified on a single line, the column number is included in the work item key:</p> <pre>Item_no1 = P750 Item_no2 = P820 Quantity1 = 1 etc.</pre>

EnterWorks Workflow Activities and Callout BIC Framework

Document Style	Description
Values only, many lines	<p>Each line has one or more values.</p> <pre>P750,P820 PrintCo 750 Inkjet Printer,PrintCo 820 Inkjet Printer 1,7</pre> <p>After import, the values are identified by the corresponding keywords. If multiple values are specified on a single line, the column number is included in the work item key:</p> <pre>Item_no1 = P750 Item_no2 = P820 Quantity1 = 1 etc.</pre> <p>If the specified work item keys are less than the number or lines in the source file, the extra lines will be ignored. If there are no specified work item keys, the default workitem keys are:</p> <pre>importa1 = first row, column 1 importa2 = first row, column 2 importb = second row importc = third row etc.</pre>

Decode Carriage Return/Line Feed Option

This option allows users to specify the symbols or characters to be decoded as carriage return/line feed. Click the **Decode CRLF encoded as:** checkbox to enable this option and the text box beside it. Enter the symbols or characters in the text box.

Import Items

The **Import Items** box displays the **Import File Key** and **Work Item Key** columns. The contents of the **Import Items** box depends on the specified document style of the delimited file to be imported.

General Details Expiration Settings **Export** **Import** Custom Action

Import Format

☒ Delimited ☐ XML ☐ Fixed

Delimiter: Comma

☐ Contains Quotes

Quote escape character: \

Document Style

Key-Values, one line DESCRIPTION,Browser,QUANTITY,1000

☐ Decode CRLF encoded as:

Import Items

Import File Key	Work Item Key
Style3ItemNumber	ItemNumberStyle3SWM
Style3RequiredCount	QuantityRequestedStyle3WSWM
Style3InvoiceNumber	InvoiceNoStyle3MappedSWM

Add Remove

OK Cancel Apply Help

The Import File Key column contains the names of the keys (as they exist in the import file) for the values you want to import. This column is enabled if you select Table with Keys, Key-Values, one-line; or Key-Values-many lines as document style. To enter a data in the Import File Key column, click **Add** and text cells will be enabled. Enter values in the text cells. Click **Remove** to delete text cells.

The Work Item Key column contains the names of the work item property table keys that you want to import. This column is enabled if you select Table without Keys; Table with Keys; Array of Values Key-Values, one line; or Key Values, many lines as document style. To enter a data in the Import File Key column, click **Add** and text cells will be enabled. Enter values in the text cells. You can also enter constants, as well as work item attributes; for example, Employee.Residence.Address. Click **Remove** to delete text cells.

Note: If you do not make selections in the Import Item box, all of the items found in the import file will be saved in the hash table.

XML

Use this option to import the XSL file as an XML file. The XSL File field is enabled when you select the XML option.

The XSL file allows the transformation of the specified XML file before it is imported to the work item properties. Type a path and file name for the XSL file or click **Browse** and a **File Open** dialog box, which allows you to select the file and path for the XSL file.

Note: When the BIC is used in a clustered environment, make sure that the folders and/or files that will be indicated are existing in and pointing to where the BIC Manager server is installed.

Use XPath

Click the **Use XPath** checkbox to locate specific portions of an XML document and to import a subset of the XML document based on the search results. To import an XML formatted file using the XPath, you must specify the XPath expressions and **Work Item Keys** in the **Import Items** box.

Below is an example of an XML document in which only the specific subset BIO will be imported using XPath:

```
<?xml version="1.0"?>
<!DOCTYPE WorkItem [
<!ELEMENT WorkItem (SYSTEM_ID,BIO,MyXMLVersion) *>
<!ELEMENT SYSTEM_ID (#PCDATA)>
<!ELEMENT BIO (AGE,INTEREST,SALARY) *>
<!ELEMENT AGE (#PCDATA)>
<!ELEMENT INTEREST (#PCDATA)>
<!ELEMENT SALARY (#PCDATA)>
<!ELEMENT MyXMLVersion (#PCDATA)>
<!ATTLIST WorkItem processVersion CDATA #REQUIRED>
]>
<WorkItem processVersion="3.1">
<SYSTEM_ID>0-34661</SYSTEM_ID>
<BIO>
<AGE>young</AGE>
<INTEREST>lawn care</INTEREST>
<SALARY>above average</SALARY>
</BIO>
<MyXMLVersion>CONSTANT1</MyXMLVersion>
</WorkItem>
```

Below is an example of an XML import configuration at the **Properties** tab:

General Details Expiration Settings Export **Import** Custom Action

Import Format

☐ Delimited ☒ XML ☐ Fixed

XSL File: Browse ... ☒ Use XPath

☐ Decode CRLF encoded as:

Import Items

XML Tag	Work Item Key
iWorkItem/BIO	bio

Add Remove

OK Cancel Apply Help

The table below shows the resulting work item properties that were created as a result of using XPath to import the specific subset BIO:

Key	Value
bio_BIO_AGE	young
bio_BIO_INTEREST	lawn care
bio_BIO_SALARY	above average
importexport_error	0

Decode Carriage Return/Line Feed Option

The Decode CRLF encoded as: option enables the text box in which you can enter the symbols or characters to be decoded as carriage return/line feed.

Import Items

The Import Items box displays the XML Tag and Work Item Key columns. The contents of the Import Items box depends on the format from which information is imported.

General | Details | Expiration | Settings | Export | **Import** | Custom Action

Import Format

☐ Delimited ☒ XML ☐ Fixed

XSL File: ☐ Use XPath

☐ Decode CRLF encoded as:

Import Items

XML Tag	Work Item Key
Workitem.age	ImportXMLAge
Workitem.po.address.city	

Note: If you do not select or specify items to import, all work item property keys and values will be written to the work item. The resulting hash table will reflect the nesting structure of the imported XML file.

The XML Tag column allows you to enter the names of the XML tags for the values you want to import. The XML tag names must be valid. You can create an XML file using nested elements. Use the dot notation to create a nested structure. To add a data, click **Add** to enable text cells and type the Import XML Tag and Work Item Key values. Click **Remove** to delete text cells.

The Work Item Key column allows you to enter the names of the work item property table keys that you want to import.

Note: After you have completed the Import or Export tab, you can click **OK** to exit the Work Item Import/Export editor and continue adding activities to your process flow.

Fixed

Use this option to import files in fixed column width and line. Selecting the **Fixed** option enables the **Data in table format** and **Ignore first row in file** checkboxes. EPX does not have the ability to parse files to determine the data structure. You must know the column and line locations of keys and data to import information from EPX.

Import Format

☐ Delimited
 ☐ XML
 ☒ Fixed

☐ Data in table format
 ☐ Ignore first line in file

☐ Decode CRLF encoded as:

Import Items

Work Item Key	Start Line	Start Column	End Line	End Column
FixDocMailFrom	2	10	2	60
FixDocMailSubject	3	10	3	100
FixDocMailMessage	5	5	6	14
FixDocMailPoemAbout...	9	1	11	

Add Remove

OK Cancel Apply Help

Data in Table Format

Select this option to specify that the data in the fixed format file to be imported is in tabular form. If the data is not in tabular form and the **Data in Table Format** checkbox is not checked, the BIC processes the file as a semi-structured document (like an e-mail message).

With this option, all rows of the specified file are processed (i.e., the start and ending line numbers are ignored). It is also assumed that each line in the file represents a row of data (i.e., data cannot span more than one line).

Ignore First Row in File

Select this option to have the Import/Export BIC ignore the key name data contained in the first row of a file.

Note: Clicking this checkbox is optional if the **Data in table format** option is not selected.

Decode Carriage Return/Line Feed Option

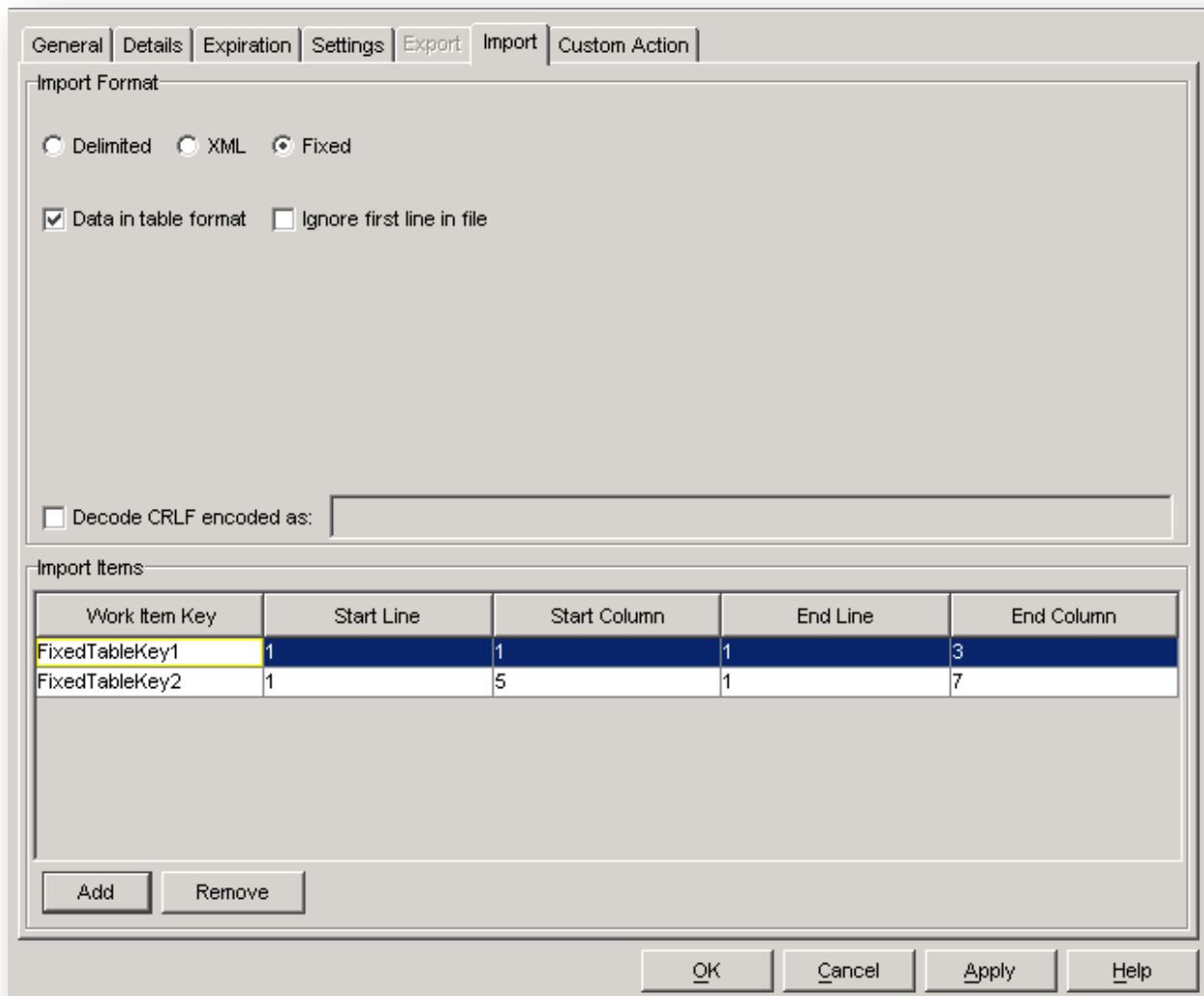
Click the **Decode CRLF encoded as:** option to enable the text box in which you can enter the symbols or characters to be decoded as carriage return/line feed.

Import Items

When you select the **Fixed** format option, the following columns are displayed in the **Import Items** box:

- **Work Item Key** – Enter the names for work item properties of the data to be imported.
- **Start Line** – Enter the line number in the file where the data to be imported into the work item property is located. The start line must be specified for each work item key to be imported.

Note: Entering data in the **Start Line** table cell is optional if the **Data in table format** option is selected.



The screenshot shows the 'Import' tab of a dialog box. The 'Import Format' section has three radio buttons: 'Delimited', 'XML', and 'Fixed' (selected). Below these are two checkboxes: 'Data in table format' (checked) and 'Ignore first line in file' (unchecked). At the bottom of this section is a checkbox 'Decode CRLF encoded as:' followed by an empty text box. The 'Import Items' section contains a table with the following data:

Work Item Key	Start Line	Start Column	End Line	End Column
FixedTableKey1	1	1	1	3
FixedTableKey2	1	5	1	7

Below the table are 'Add' and 'Remove' buttons. At the bottom of the dialog are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

- **Start Column** – Enter the numeric value for the column position of the first character of data to be imported into the work item property is located. The starting column must be specified for each work item key to be imported.
- **End Line** – Enter the numeric value for the line number when the BIC should stop reading import data.

Note: Entering data in the Start Line table cell is optional if the Data in table format option is selected.

- **End Column** – Enter the numeric value for the column position where the BIC must stop reading import data.

To add values in the appropriate text cells, click **Add** to enable text cells and enter data in the text cells. Click **Remove** to delete text cells.

Export Tab

The **Export Tab** allows you to configure the Work Item Import and Export BIC to export information from the work item it receives and save the information in delimited, fixed, or XML formats. After you select a format, the BIC provides you access to only those options required to configure the export file.

General | Details | Expiration | Settings | **Export** | Import | Custom Action

Export Format

☒ Delimited ☐ XML ☐ Fixed

Delimiter: Comma

☐ Use Quotes

Quote escape character: \

Document Style

Table with keys

DESCRIPTION,QUANTITY
Browser,1000

☐ Encode CRLF as:

☐ Include keys in export ☐ Include system properties in export

Export Items

Export File Key	Value

Add Remove

OK Cancel Apply Help

Delimited

Use this option if you want to set the separator for values in files to be exported. The **Delimiter** dropdown list is enabled when you select the **Delimited** option.

Export Format

☒ Delimited ☐ XML ☐ Fixed

Delimiter: Comma

☐ Use Quotes

Quote: Semicolon

Document Style: Table with keys

Preview: DESCRIPTION,QUANTITY
Browser,1000

☐ Encode CRLF as:

☐ Include keys in export ☐ Include system properties in export

Export Items

Export File Key	Value
biosalary	%biographical.salary%
bioage	%biographical.age%

Add Remove

OK Cancel Apply Help

Delimiter

From the **Delimiter** dropdown list, select the field delimiters for files to be exported (e.g., Comma, Tab, Semicolon, Space, or Other Delimiter). If you select **Other Delimiter**, enter the delimiter in the text box to the right of the **Delimiter** text box. If one or more of the work item property values contains the selected delimiter, you can enclose the values the BIC exports in double quotes.

Document Style

The **Document Style** box allows user to select a document style when exporting a file in a delimited format. This enables the BIC to format the output text file according to the chosen document style. The document style formats are similar to what is described under the **Import Delimited** section.

Include System Properties in Export

The Include system properties in export checkbox is enabled when selecting the **Delimited** option. If this checkbox is selected, the system properties are included in the delimited file exported from EPX.

The screenshot shows the 'Export' tab of a configuration window. The 'Export Format' section has three radio buttons: 'Delimited' (selected), 'XML', and 'Fixed'. Below this, there is a 'Delimiter' dropdown set to 'Comma', a 'Use Quotes' checkbox (unchecked), and a 'Quote escape character' text box containing '\\'. The 'Document Style' section has a dropdown set to 'Table with keys', which is expanded to show a preview of a table with two rows: 'DESCRIPTION,QUANTITY' and 'Browser,1000'. Below this, there is an 'Encode CRLF as:' checkbox (unchecked) and a text box. At the bottom of this section are two checkboxes: 'Include keys in export' (unchecked) and 'Include system properties in export' (unchecked). The 'Export Items' section is a table with two columns: 'Export File Key' and 'Value'. It is currently empty. Below the table are 'Add' and 'Remove' buttons. At the very bottom of the window are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

Encode Carriage Return/Line Feed Option

Click the **Encode CRLF as:** option to enable the text box to get the symbols or characters to be encoded as carriage return/line feed.

Export Items

The **Export Items** box changes based on the format to which information is exported. When you select **Delimited**, you can configure the **Export File Key** and **Value** text cells. Click **Add** to enable text cells in which you can type values. Click **Remove** to delete text cells.

The **Export Items** box contains the following:

- **Export File Key** – Enter the names of the keys for the values you want to export.

EnterWorks Workflow Activities and Callout BIC Framework

- **Value** – enter the names of the work item property table keys that you want to export. All key names must be enclosed in % (percent signs); for example, %address.street%. Constants, wildcards (*), and EPX work item attributes can also be used as values. In using wildcards, the output to export file depends on four cases shown in the table below.

Note: A blank Export Items field produces the same result as Case 1. Cases 1 and 4 ignore the Export File Key input, while cases 2 and 3 use it to replace the wildcard keyword in the exported key names.

Case	Description	Example	Export file output
1	Asterisk is the only argument.	*	All workitem properties
2	Asterisk is the starting argument.	*text	Items with key names ending in “text”
3	Asterisk is the ending argument.	text*	Items with key names beginning in “text”
4	Enclosed within asterisks	*text*	Items with key names containing “text”

To export all workitem properties, set a single asterisk (*) as the only argument in the Value table.

To export items with key names ending in “text”, put an asterisk in the beginning of the argument.

Consequently, put an asterisk in the end of the argument if the key names should begin in text. For items with key names containing text, enclose the text argument within asterisks. Refer to the table above for examples of using wildcards as export item values.

General Details Expiration Settings **Export** Import Custom Action

Export Format

☒ Delimited ☐ XML ☐ Fixed

Delimiter: Semicolon

☐ Use Quotes

Quote escape character: \

Document Style

Table with keys

DESCRIPTION,QUANTITY
Browser,1000

☐ Encode CRLF as:

☐ Include keys in export ☐ Include system properties in export

Export Items

Export File Key	Value
CustomerName	%CustomerName%
ServerName	%sys.server.systemname%
ValueConstant	CONSTANTVALUE
BioSalary	%biographical.salary%

Add Remove

OK Cancel Apply Help

Note: If you do not specify Export File Key and Value data, all the work item property keys and values will be written to the export file.

XML

Selecting the XML option activates the XSL text box and the **Browse** button.

General | Details | Expiration | Settings | **Export** | Import | Custom Action

Export Format

☐ Delimited ☒ XML ☐ Fixed

XSL File: Browse ...

DTD Generation: ☒ Internal ☐ External ☐ None

Root element name:

☐ Encode CRLF as:

☐ Include keys in export ☐ Include system properties in export

Export Items

XML Element	Value

Add Remove

OK Cancel Apply Help

XSL File

The Work Item Import and Export BIC uses the structure of the work item hash table as the default structure for the XML document that you export. You can directly type a path and file name for the XSL file that the BIC uses to format or transform the exported file. You can also click **Browse** and select the file and path for the XSL file in the <EPX>\bin\agents\<hostname> directory on the **File Open** dialog box.

Note: When the BIC is used in a clustered environment, make sure the folders and/or files that will be indicated are existing in and pointing to where the BIC Manager server is installed.

Below is an example of an XSL style sheet that can be used to transform an XML file into an HTML-formatted file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/WorkItem/Personal">
```

```

<html>
<body>
<h2>My Sample StyleSheet</h2>
<table border="1" width="250">
<tr>
<td align="center" colspan="2">
<B>Personal Information</B></td>
</tr>
<tr>
<th align="center">Field</th>
<th align="center">Value</th>
</tr>
<tr>
<td>FirstName</td>
<td><xsl:value-of select="FirstName"/></td>
</tr>
<tr>
<td>LastName</td>
<td><xsl:value-of select="LastName"/></td>
</tr>
<tr>
<td>MiddleInitial</td>
<td><xsl:value-of select="MiddleInitial"/></td>
</tr>
<tr>
<td>DateOfBirth</td>
<td><xsl:value-of select="DateOfBirth"/></td>
</tr>
<tr>
<td>Sex</td>
<td><xsl:value-of select="Sex"/></td>
</tr>
</table>
<br></br>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Below are the contents of an exported HTML file that was formatted using XSL file shown above:

My Sample StyleSheet

Personal Information	
Field	Value
FirstName	Joe
LastName	Smith
MiddleInitial	M
DateOfBirth	09/20/1972
Sex	Male

Here is another example of an XSL style sheet that can be used to transform an XML file into another XML-formatted file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<Product>
<SPSC_Code>
<xsl:value-of select="WorkItem/SPSC_Code" />
</SPSC_Code>
<Supplier_ID>
<xsl:value-of select="WorkItem/Supplier_ID" />
</Supplier_ID>
<Long_Desc>
<xsl:value-of select="WorkItem/Long_Desc" />
</Long_Desc>
<Expiration_Date>
<xsl:value-of select="WorkItem/Expiration_Date" />
</Expiration_Date>
</Product>
</xsl:template>
</xsl:stylesheet>
```

Below are the contents of an exported XML file that was formatted using XSL file shown above:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Product>
<SPSC_Code>11112222</SPSC_Code>
<Supplier_ID>aaa1</Supplier_ID>
```

```
<Long_Desc>ddd1</Long_Desc>  
<Expiration_Date>08/19/2003</Expiration_Date>  
</Product>
```

DTD Generation

Specify the type of DTD Generation you want to use in the XSL file. The following are the DTD Generation options:

- **Internal** – selecting this option will embed the DTD with the XML file.
- **External** – select this option if you want to create a separate DTD file.
- **None** – select this option if you do not want to create a DTD file.

Root Element Name

The default Root Element Name is WorkItem because this is where the work item property keys and values are to be exported and contained. The default root element tag name can be overridden by entering a value in the Root element name field.

Include System Properties in Export

The Include system properties in export checkbox is enabled only when you select XML as the output format. If this checkbox is selected, the system properties are included in the XML file exported from EPX.

The 'Export' dialog box is shown with the following settings:

- Export Format:**
 - ☐ Delimited
 - ☒ XML
 - ☐ Fixed
- XSL File:** [Empty text box] **Browse ...**
- DTD Generation:**
 - ☐ Internal
 - ☐ External
 - ☒ None
- Root element name:** [Empty text box]
- Encode CRLF as:** ☐ [Empty text box]
- Include keys in export:** ☐
- Include system properties in export:** ☒

Export Items:

XML Element	Value

Add **Remove**

OK **Cancel** **Apply** **Help**

Encode Carriage Return/Line Feed Option

Click the **Encode CRLF as:** option enables the text box to get the symbols or character to be encoded as carriage return/line feed.

Export Items

For the XML export format, **Export Items** allows you to conditionally choose the work item properties to export and its corresponding XML tag. You can add more than one export item. If no export items are specified, all work items will be exported. The resulting XML file reflects the nesting structure of the work item properties.

Click **Add** and enter the export item. Click the **Remove** button to delete an export item.

General Details Expiration Settings **Export** Import Custom Action

Export Format

☐ Delimited ☒ XML ☐ Fixed

XSL File: Browse ...

DTD Generation: ☒ Internal ☐ External ☐ None

Root element name:

☐ Encode CRLF as:

☐ Include keys in export ☒ Include system properties in export

Export Items

XML Element	Value
BIO.SALARY	%biographicaldata.salary%
BIO.AGE	%biographicaldata.salary%
MyXMLVersion	CONSTANT1
SYSTEM_ID	%epi.workitem.version.uid%

Add Remove

OK Cancel Apply Help

The Export Items box contains the following columns:

- **XML Element** – Enter the names of the XML tags for the values you want to export. The XML tag names must be valid. You can specify nested tags by using slashes “/” to separate the parent node with the child node.
- **Value** – Enter the names of the work item property table keys you want to export. You must enclose all of the key names in percent signs (%), e.g., %address.street%.

Constants, wildcards (*), and EPX work item attributes can also be used as values. In using wildcards, the output to export file depends on four cases shown in the table below.

Note: A blank Export Items field produces the same result as Case 1. Cases 1 and 4 ignore the Export File Key input, while cases 2 and 3 use it to replace the wildcard keyword in the exported key names.

EnterWorks Workflow Activities and Callout BIC Framework

Case	Description	Example	Export file output
1	Asterisk is the only argument.	*	All workitem properties
2	Asterisk is the starting argument.	*text	Items with key names ending in "text"
3	Asterisk is the ending argument.	text*	Items with key names beginning in "text"
4	Enclosed within asterisks	*text*	Items with key names containing "text"

To export all workitem properties, set a single asterisk ('*') to be the only argument in the Value table.

To export items with key names ending in "text", put an asterisk in the beginning of the argument.

Put an asterisk in the end of the argument if the key names should begin in text. For items with key names containing text, enclose the text argument within asterisks. Refer to the table above for examples of using wildcards as export item values.

Fixed

Selecting the Fixed option enables the Include keys in export checkbox.

General | Details | Expiration | Settings | **Export** | Import | Custom Action

Export Format

☐ Delimited ☐ XML ☒ Fixed

☐ Encode CRLF as:

☐ Include keys in export ☒ Include system properties in export

Export Items

Export File Key	Value	Starting Column

Add Remove

OK Cancel Apply Help

Include Keys in Export

The Include keys in export checkbox is available only when you select **Fixed** as export format. The Clicking this checkbox includes hash table keys in the delimited file exported from EPX.

Note: The Include system properties in export checkbox is disabled when you select other export formats because only the Fixed option allows items with Export File Key, Value, and Starting Column to be exported. If you want to export the system properties, enter the appropriate values in the text cells provided for the Export Items.

Encode Carriage Return/Line Feed Option

Clicking the Encode CRLF as: option enables the text box to get the symbols or characters to be encoded as carriage return/line feed.

Export Items

The Export Items box changes based on the format to which information is exported. When you select **Fixed** format, the following columns appear:

EnterWorks Workflow Activities and Callout BIC Framework

- **Export File Key** – Enter the names of work item properties you want to export. You can enter EPX System keys and constants.
- **Value** – enter the names of the work item property table keys that you want to export. You can also type constants and EPX work item attributes. You must enclose the key names in the % character, for example, %sys.server.name%.

Fixed format does not accept wildcards as export item values.

- **Starting Column** – enter the numeric value for the position that the exported value's first character will occupy in the export file. The starting column must be specified for each file key.

Click **Add** to enable text cells. Click **Remove** to delete text cells.

General Details Expiration Settings **Export** Import Custom Action

Export Format

☐ Delimited ☐ XML ☒ Fixed

☐ Encode CRLF as:

☒ Include keys in export ☒ Include system properties in export

Export Items

Export File Key	Value	Starting Column
CustomerName	%CustomerName%	1
BioSalary	%Biographicaldata.salary%	40
SomeConstant	CONSTANT_VALUE	80

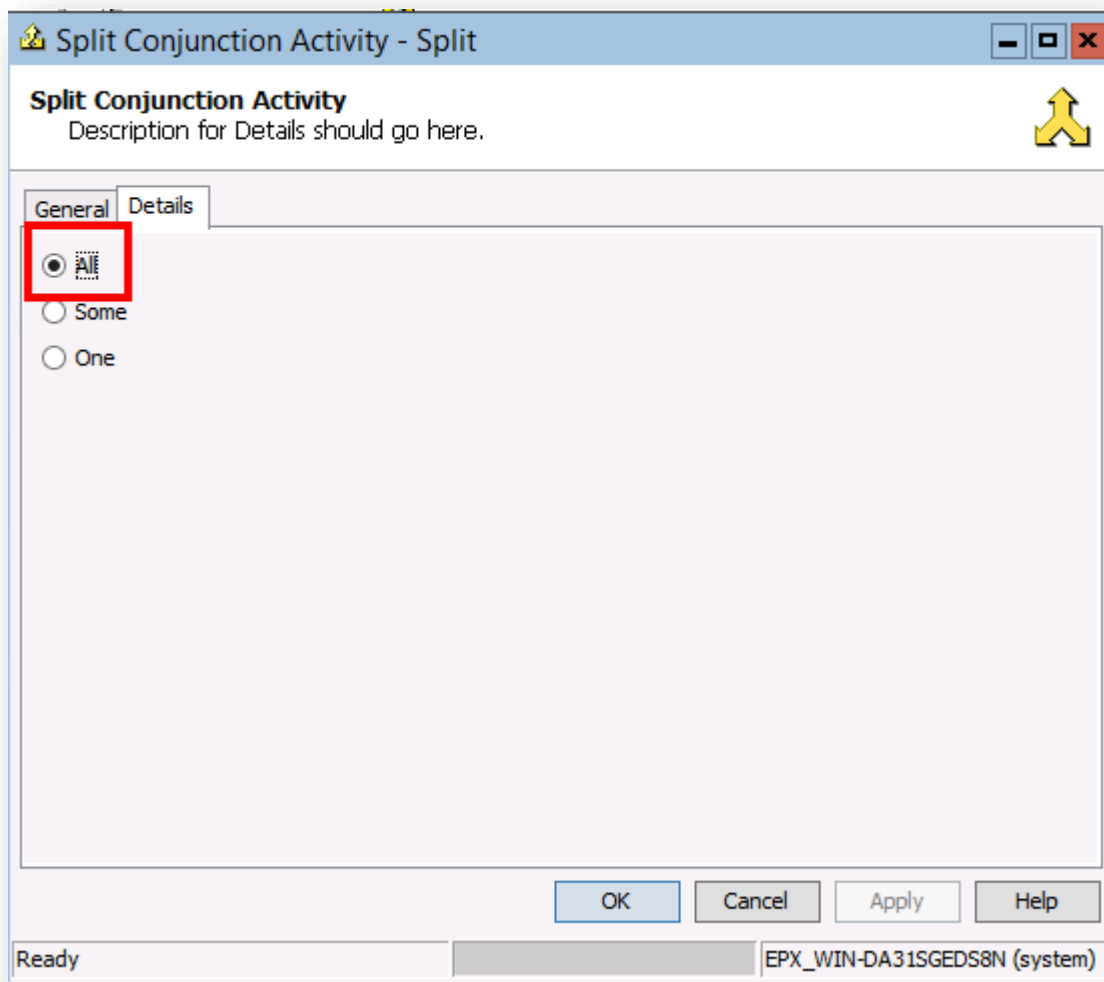
Add Remove

OK Cancel Apply Help

Split Activity



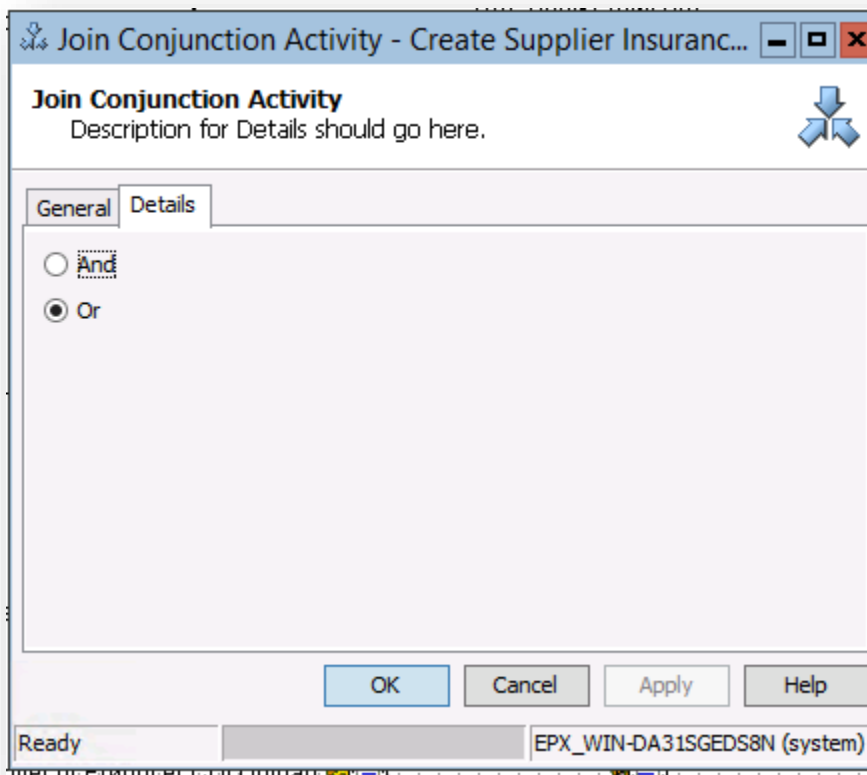
The Split activity is used to split a work item so it may follow parallel paths, such as facilitate reviews/updates by multiple groups concurrently. The Split activity has options for splitting the work item to all paths, some or just one. Only the All path should be used. If there is a need to perform a conditional split (i.e., only route the work item to some of the possible paths), the Decision Point Activity should be used instead. If the split paths will be interacting with users (i.e., contain manual steps), the Decision Point Activity should be used to split the work items as well.



Join Activity



The Join Activity provides the means for diverging paths to be recombined into a single path.



The Join activity can be configured with two behaviors:

- OR – the work item is sent to the next activity immediately, even if it was split.
- AND – all versions of the work item must be sent to this activity before all of them are sent to the next activity. Usually when AND would be used, the appropriate activity would be to use a Merge to recombine the work items (the AND Join does not combine them).

Decision Point Activity



The Decision Point Activity provides the ability for work items to take different paths in the work item and to possibly split the work item to follow parallel paths. The activity must be configured to have at least one condition with a destination for each. Each condition can check the value of properties in the work item. Each condition can have one or more destinations specified. If multiple destinations are specified, the work item will be split. If more than defined one condition is met, the work item will also be split and sent to the corresponding destination for each matching condition.

A Decision Point Activity, or DPA, intelligently routes work items through a process flow based on both the rules you configure and the information in a work item.

Before creating a DPA, you need to know:

- What actions you want the DPA to take.

Base the DPA rules on variables in your work items. For example, a DPA might determine which activity to send a biographical data work item to, based on whether an age variable in the work item matches a value specified in one of the DPA's rules.

- What work item properties are available for the DPA in the process flow or subflow.

The DPA can retrieve the properties within a work item and examine its Stored Procedure execution.

Finding Information for the DPA

The DPA acts on the content of a work item. As the DPA's designer, you need to find the information required by the DPA that is being configured. This information can come from a Work Item Viewer in a manual activity or from a BIC. If a process modeler maintains the work item type metadata and ensures that viewers and automatic activities are consistent with it, then it can be used as a source when defining the rules within the DPA.

Configuring DPAs

To configure a DPA in a flow, open the flow editor to the Process Flow Modeler tab and proceed as follows:

1. Place a DPA in the flow.
2. Right-click the DPA icon and click **Properties** in the shortcut menu to launch the **Decision Point Activity** editor.
3. On the **General** tab, type a name for this activity in the **Name** box and a description (optional) in the **Description** box. If you do not specify a name, EPX assigns a default name such as **Decision Point Activity (1)** or **Decision Point Activity (2)**.
4. Click the **Details** tab. If needed, select a new priority for work items. Refer to "Using a BIC in a Flow," on page 86 for more information.

Note: Since DPAs are used to route work items, they cannot be starting points in a flow. Another activity must precede a DPA and send work items to the DPA.

5. Click the **Settings** tab. The settings table displays any rules previously created for this DPA. To edit an existing rule, right-click the rule and click **Open** in the shortcut menu. The **Rule Properties** dialog displays the rule you selected.

To create a new rule, right-click in the settings table and click **Add** in the shortcut menu. The **Rule Properties** dialog box displays.

Note: You cannot save a rule unless the DPA is connected to at least one other activity on the flow canvas.

Click the **External Properties** tab to allow the system to call out stored procedures defined in the database to retrieve properties to be tested within a DPA condition.

6. When you have finished configuring this activity's properties, click **OK** to close the DPA editor and save the settings.

Configuring Rules for DPAs

You can configure **Simple**, **Advanced**, and **Otherwise** rules for DPAs. **Simple** and **Advanced** rules permit you to establish an if-then condition that instructs the DPA to look at the work item and send it to another activity depending on the rule and on what is in the work item.

Note: If more than one condition is met, then the work item will be sent to multiple destinations. A work item version being sent to multiple destinations is, in effect, splitting the work item. Those split work item versions can be subsequently combined using the Merge Activity.

If none of the conditions specified in the rules are met, the DPA will refer to the **Otherwise** rule if it is defined. If it is not defined, the work item will fail to be sent from the activity preceeding the DPA. If this is a manual activity, an error will be displayed on the TaskManager from which the work item was sent. If this is an automatic activity, the work item will remain at that activity and be flagged with an error.

- **Simple** rules — allow you to specify basic if-then conditions. For example, if the value 'young' is selected in the work item, send the work item to the approval activity.
- **Advanced** rules — allow you to specify more complex if-then conditions. For example, if the value 'young' is selected in the work item and the value >\$30,000 is also selected, then send the work item to the approval activity.
- **Otherwise** rule — allows you to specify a rule that applies if none of the other if-then conditions are met. In each DPA, only one **Otherwise** rule can be defined.

Configuring Simple Rules for DPAs

To set Simple rules for a DPA:

1. In the Rule Properties dialog, select the **Simple** option.

To set **Simple** rules, select the appropriate data type in the **Value Type** list.

For datetime data type, the dates should be in four-digit year format. Hence, use the datetime format mm/dd/yyyy hh:mm.

Other recognized data types are number and string. Boolean data type, however, merely supports the case-insensitive strings 'True' and 'False' values.

2. In the **Field Name** box, type the name of the work item field. For example, type `biographicaldata.age` to specify the age field.
3. In the **Operator** box, select the comparison operator. The list of operators will be based on the data type selected.
4. In the **Value** box, type the value of the work item field. For example, type 'young'.

Note: The value entered must be a literal of the appropriate value type. To compare two fields, the simple rule can be used by having the property referenced in the **Value** field to be enclosed within '%'. For example, '%biographicaldata.age%'.

If

Type: ☒ Simple ☐ Advanced ☐ Otherwise

Value Type: String

Field Name: biographicaldata.age

Operator: =

Value: 'young'

Then Send To


Available: Middle Activity, Old Activity

Selected: Young Activity

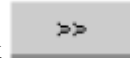
Buttons: <>, >>, <<

Error Messages

OK Cancel Validate Clear

5. Click an activity in the **Available** box and click  to move the activity to the **Selected** box. The DPA is now configured to send the work item to this activity.

To move all activities from the **Available** box to the **Selected** box, click



To move all activities from the **Selected** box to the **Available** box, click



6. Click **Validate** to verify that your rules contain no errors. Any errors display in the **Error Messages** box.
7. To save the rule, click the **OK** button.

Setting an Otherwise Rule for DPAs

Creating an **Otherwise** rule tells the DPA where to send a work item if the conditions of other rules are not met.


1. Open the **Rule Properties** dialog as described in, “Configuring Simple Rules for DPAs,” on page 92.
2. In the **Rule Properties** dialog, select the **Otherwise** option.

Note: You can create only one otherwise rule for each DPA.

The screenshot shows a dialog box titled "If". It has three main sections:

- Type:** Three radio buttons are present: "Simple", "Advanced", and "Otherwise". The "Otherwise" button is selected.
- Then Send To:** This section contains two lists. The "Available" list on the left contains the item "Delivery". The "Selected" list on the right contains the item "Approval". Between these two lists are three buttons: "<>", ">>", and "<<".
- Error Messages:** A text area at the bottom of this section contains the message "Rule is valid" in blue text.

At the bottom of the dialog box are four buttons: "OK", "Cancel", "Validate", and "Clear". The "Validate" button is highlighted with a dashed border.

3. Click an activity in the **Available** box and click  to move the activity to the **Selected** box. The DPA is now configured to send the work item to this activity.
4. To save the rule, click the **OK** button.

Setting Advanced Rules for DPAs

Using the DPA's **Advanced** option feature, you can extend the rule to perform two or more comparisons simultaneously. For example, you can set a rule that matches not just age criteria, but also matches employment status.

1. Open the Rule Properties dialog as described in, "Configuring Simple Rules for DPAs," on page 92.

2. In the Rule Properties dialog, select the Advanced option.
3. In the text box, type the Advanced rule.

- For string fields, prepend with **String:**
- For date fields, prepend with **Date:**
- For datetime fields, prepend with **Datetime:**
- For number fields, prepend with **Number:**
- For boolean fields, use string field reference with values '**True**' and '**False**'.

4. Combine comparisons with a pipe (|) if only one of the given conditions must be met, or with an ampersand (&) if both conditions must be satisfied.

For example, to find young or employed people in a biographical data work item, type:

```
(String:biographicaldata.age='young') |
(String:biographicaldata.status='employed')
```

To find young people with a salary of more than \$50,000 on July 23, 2003, type:

```
(String:biographicaldata.age='young') & (Number:salary>50000)
&
(Date:presentdate=07-23-2003)
```

To find employed people with a salary of more than \$50,000 on July 23, 2003 at 01:00 am, type:

```
(String:biographicaldata.status='employed') & (Number:salary>
50000)
& (Datetime:presentdatetime=07/23/2003-01:00-AM)
```

Note: Place single quotes around strings.

General Guidelines for Establishing Rules

To create **Advanced** rules you can use numbers, strings, collections, datetimes and dates. Operators such as: +, -, *, /, =, >, <, >=, <=, <>, and = are valid. You can also use functions in your **Advanced** rules.

Valid Formats

For:	Examples
Number	5 5.0 .5 4.5555
String	'how' 'adlkjljaf343' 'John Doe' 'B-1'

Strings must start with a letter or a digit but can be followed by none or any number of other letters and digits, blank spaces, hyphens (-), number signs (#), and percent signs (%).

Date	Use the format mm-dd-yyyy. The date 09-11-1997 is valid.
Collection types	4 'how' 09-09-1997 Valid collection types are numbers, expressions, or dates. Collections can also be strings.
Datetime	Use the four-digit year format mm/dd/yyyy-hh:mm-AM. For example, 09/11/1997-08:30-AM

Variables

You can also use variables (work item properties) in the expression. To use a variable in an expression, make sure that you prefix the variable name by its type.

For example, if the variable is a number, type:

Number:varName

For strings and dates, type:

String:varName and Date:varName

For a datetime type:

Datetime:expiredDate or Datetime:sendDate

Operators

For	Possible Values
Number	+ - * / = > < >= <= <>
String	+ = <>
Date	> < = <>
Datetime	> < = <>
Comparing logical expressions	& (Ampersand for and) (Pipe for or)

The set up of advanced rules with operators must follow the correct syntax. Parameters (data type prefix and variable pair) should always be placed within parentheses to pass validation. For example:

Number:a= (Number:b) + (Number:c) -
(Number:d) * (Number:e) / (Number:f)

Usage

Examples of advanced rule operators usage:

For	Examples
Number	Number:biographical.age=((4*(4-2))
Dates	Date:biographical.bdate<>09-09-1990
Datetime	Datetime:present>01/23/2003-08:00-AM

Functions

Available functions are: max, absolute, length, and date. Place functions on the right-hand side of the expression. For example:

```
abs (3) =max (3, 5)
```

This example is not valid, because the function abs (absolute) is to the left of the equal sign (=).

Another example:

```
5=max (5, 5)
```

This example is valid, because the function, max, is to the right of the equal sign (=).

Function	Code	What it does	Arguments
Max	max	Compares two numbers to see which is greater	Requires two numbers separated with a comma: Number,Number
Absolute	abs	Finds the absolute value of a number	Requires one number only
Length	length	Finds the number of characters in a string	Enter the string within single quotes
Date	date	Determines the current system date	Takes no arguments
Datetime	datetime	Determines the current system date and time	Takes no arguments

Note: When changing the target activity for a previously-configured Decision Point Activity, the following steps must be performed to avoid losing the details of the condition:

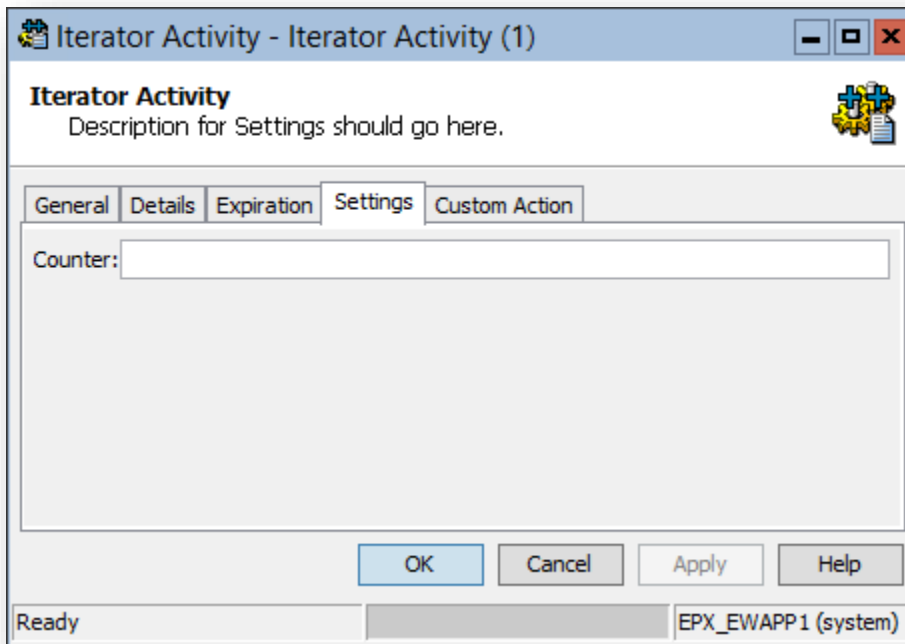
1. Add the new activity to the canvass.
2. Connect the Decision Point Activity to the new activity.

3. Edit the Decision Point Activity and change the routing of the appropriate conditions from the activity being replaced with the new activity.
4. Save the Decision Point Activity changes.
5. Sever the connection between the Decision Point Activity and the old activity.
6. Reconnect the new activity to its appropriate destination.
7. Save the workflow.

Iterator Activity



An Iterator Activity counts the number of times that a work item passes through a loop within a flow by incrementing the value associated with a key that you specify. Among the properties you can set for an Iterator Activity are its work item priority and expiration period, and the work item property name for its counter value.



Iterator Activities count the number of times a loop in a flow is completed, increasing the iteration count by one each time. Typically, this activity is used with a DPA, which routes the work item to the appropriate activity in the process flow based on the count. Use the DPA immediately after the Iterator Activity. A DPA allows multiple Transitions out and can send a work item back into a previous activity in the flow or to the next destination.

For example, after each time the Iterator Activity increases the iteration number for a work item, it sends the work item to the DPA. If the work item has completed enough iterations, then the DPA then sends the work item to the next destination. If the work item has not completed enough iterations, then the DPA returns the work item to a previous activity in the flow.

Configuring Iterator Activities

To configure this activity:

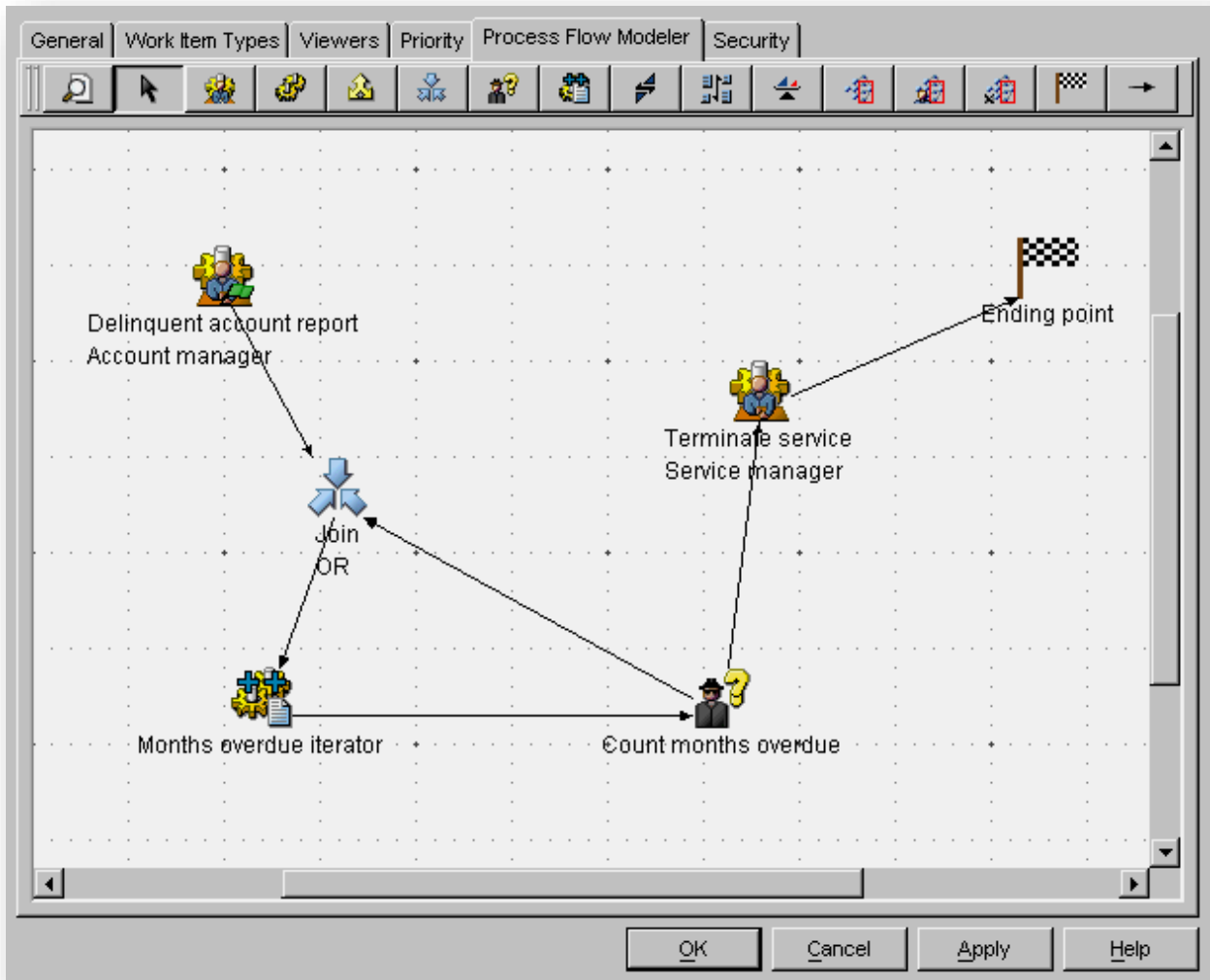
1. Place an Iterator Activity in the flow, right-click its icon, and click Properties in the shortcut menu.
2. On the General tab, type a name for the activity in the Name box and a description (optional) in the Description box.
3. Click the Details tab. If needed, select a new priority for work items.

Note: This activity cannot be a starting point.

4. Click the Settings tab.
5. In the Counter box, type the name of the counter to be incremented. If the counter does not exist, it will be created with a value of 1.
6. Click **OK**.

Using Iterator Activities

This activity iterates a counter by one for each work item version that it receives.



In the process flow above, the Iterator Activity initially receives all work items from the first activity, by way of an Or Join Activity. The Iterator Activity creates a counter key and gives it a value of 1. The work item version is then passed to a DPA, which reads the counter's value and, if the counter value has reached the level defined in the DPA rule, sends the work item to the final activity in the flow. If the target value has not yet been met, the work item is sent back to the Or Join Activity, creating a loop. The work item passes through the Iterator Activity again, which iterates the counter value by one. This repeats either until the target counter value is reached, or the work item is purged.

Merge Work Items Activity



The Merge Work Items Activity recombines split versions of the same work item back into one work item, combining their work item properties. There are some limitations when managing work items that have been split. If the split versions of the work item have updates made to the same properties, and those values are different, when the work items are merged, those differing values will be combined into a pseudo-array. This pseudo-array is not consistent, making it very difficult to properly handle such properties. Therefore, having merged work item versions produce the pseudo-arrays should be avoided. This can be accomplished in several ways:

1. Ensure that each parallel path updates different work item properties. For example, if parallel reviews are being conducted by different groups, name the properties set in those reviews to reflect the group (e.g., qaApproval=Yes/No, salesApproval=Yes/No, etc.)
2. Ensure the properties that may have had different values are reset to the same value in each path prior to the merge. For example, each path may include multiple steps with branching or even looping and work item properties will need to be used to control that. But once that processing has completed and the work item versions are ready to be merged, those properties are no longer needed. A Set Properties Callout BIC activity can be configured to precede the Merge activity to reset all of those properties so that when the work item versions are merged, each version will have the same values for those properties resulting in a single property (vs. the pseudo array properties) in the merged work item.

Work Item Priority Activity



The Work Item Priority Activity allows for the priority of the work item to be set. This feature is not used in current Enable workflow implementations, so this BIC should not be used.

Load Balancing Activity



The Load Balancing Activity provides the ability to route work items to different destinations based solely on load. The Load Balancer Activity operates strictly on a round-robin basis, so it has limited use since work items may pile up on one of the destinations while the others are processed and then remain idle.

Subflow Activity



The Subflow Activity provides the ability to pass a work item from a process flow or subflow to a designated subflow. Once the work item completes the processing in that subflow, it is advanced to the next activity following the Subflow Activity.

Personal Subflow Activity



The Personal Subflow Activity provides the ability to pass a work item from a process flow or subflow to a personal subflow, which mimics an interactive wizard. Due to the way interactive workflows are now integrated with Enable, there is not much call to implement personal subflows so this activity should not be used.

Remote Subflow Activity



The Remote Subflow Activity provides the ability for a work item to be passed to a subflow in a registered remote EPX server. Having a distributed EPX environment is not a standard configuration nor will it ever be used, so this activity should not be used.

End Activity



The End Activity provides a termination point for a workflow. Once a work item has reached an End activity, it is no longer active in the workflow and the Enable record anchored to the work item is automatically released from the workflow lock. A workflow may have more than one end activity.

Callout BIC Framework

The workflow Callout BIC provides a standard method of implementing and configuring processing within an EPX workflow. The Callout BIC allows the data modeler to specify the fully qualified path for a Java class that implements the Callout BIC interface. Once the class has been specified, the Callout BIC presents the user with a form containing the following information:

- Detailed description of what processing the class performs.
- List of input parameters including descriptions (as tool tips on the parameter names) for each with a corresponding input field to specify the value for each parameter. The value can be literal text or numeric, a reference to a work item property, or the combination of both. For example, if the class includes a property named “resultsLogfile” with a description “Fully-qualified name of the file to be generated” and a work item property “baseDirectory” defines

the directory where the file is to be placed, the value for the property might be set to: “%baseDirectory%/MyResultsFile.txt” (without the quotes). If a parameter is required, it will include an asterisk at the end of the parameter name. If a parameter has a default value defined, it will populate the field any time the field is cleared.

- List of output parameters with descriptions (as tool tips on the parameter names) with the option to override the property names.

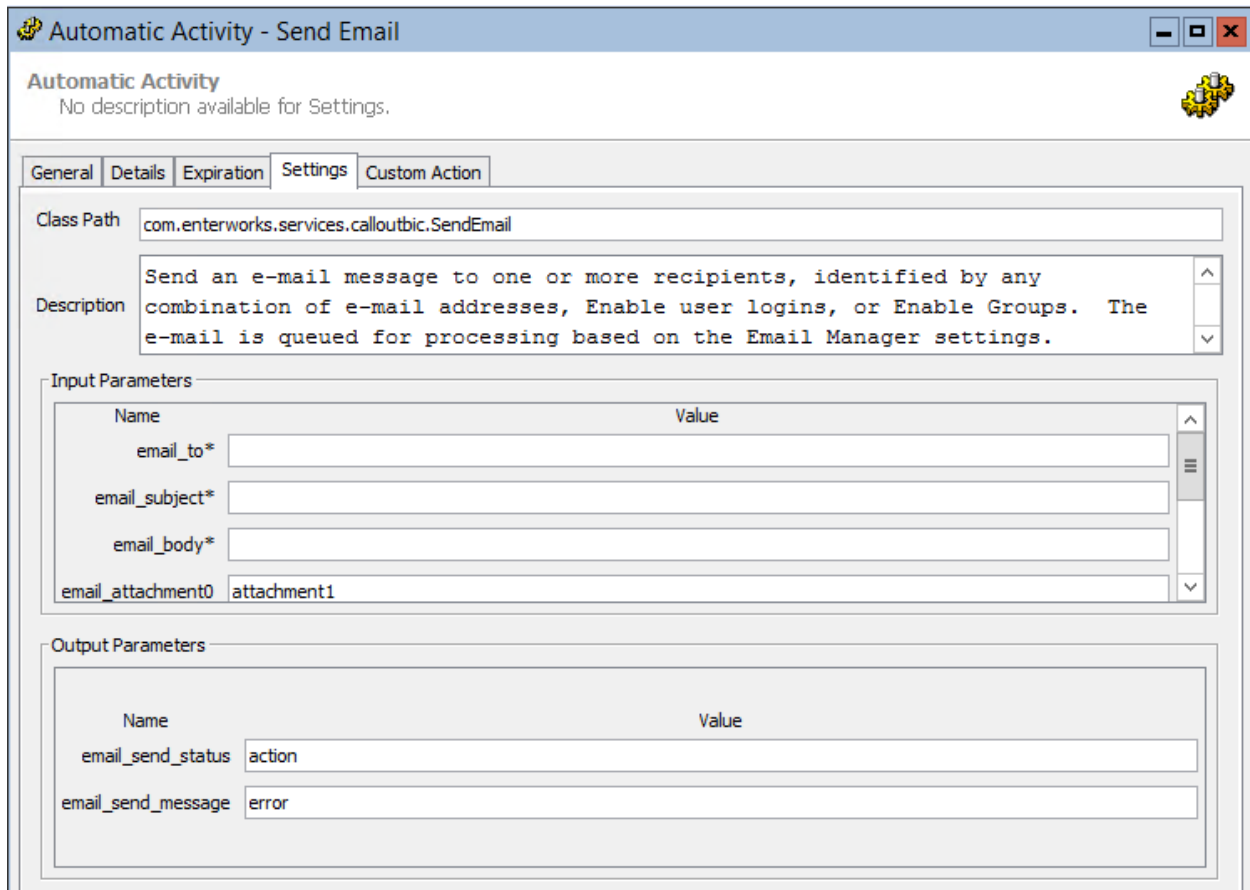
When a Callout BIC class is entered in the Settings tab, it will display the detailed description for the class along with the list of input and output properties for the class and detailed descriptions for each. The example form below shows a possible Callout BIC Class that would be used to ensure the record associated to the work item has the appropriate images (by context) linked to it (the shaded cells are input fields in the form):

Processing Class:	com.enterworks.services.callout.ValidateLinkedImages	
Description:	Verifies the repository record has at least the specified minimum but no more the maximum specified linked image for each specified Context. Up to 5 image contexts can be defined	
Input Parameters		
Property	Value	(Tool Tip)
imageContext1	Main	Identifies the Image Context name for context 1
minImageCount1	1	Identifies the minimum number of images that must be linked for imageContext1. The default value is 1.
maxImageCount1	1	Identifies the maximum number of images that can be linked for image Context1. The default value is 1.
imageContext2	Logo	Identifies the Image Context name for context 2
minImageCount2	1	Identifies the minimum number of images that must be linked for imageContext2. The default value is 1.
maxImageCount2	3	Identifies the maximum number of images that can be linked for image Context2. The default value is 1.
...		
imageContext5		Identifies the Image Context name for context 5
minImageCount5		Identifies the minimum number of images that must be linked for imageContext5. The default value is 1.
maxImageCount5		Identifies the maximum number of images that can be linked for image Context5. The default value is 1.
Output Parameters		
Property	Override Property	(Tool Tip)

isValid		Set to Yes if the required image contexts are linked, otherwise set to No
errorMessage	linkedImageErrors	Detailed error message if the record is missing required linked images

When executed, the input parameters are passed to the designated class along with standard parameters (e.g., work item id, repository ID or saved set ID, etc.). The class returns true if the operation was successful or false if it failed (the reason for failure will be returned as a text message). The Callout BIC class also creates or updates the output parameters with the results of the processing. If the processing failed, a BIC Exception will be thrown using the error message returned by the class. It is up to the EPX workflow to either handle the error (with the Callout BIC Send work item on error enabled) or to leave the work item stuck at the activity with an error.

The following is a screen shot of the SendEmail Callout BIC Settings tab:



Automatic Activity - Send Email

Automatic Activity
No description available for Settings.

General Details Expiration **Settings** Custom Action

Class Path: `com.enterworks.services.calloutbic.SendEmail`

Description: Send an e-mail message to one or more recipients, identified by any combination of e-mail addresses, Enable user logins, or Enable Groups. The e-mail is queued for processing based on the Email Manager settings.

Input Parameters

Name	Value
email_to*	
email_subject*	
email_body*	
email_attachment0	attachment1

Output Parameters

Name	Value
email_send_status	action
email_send_message	error

The base class for the Callout BIC class will have a library of support methods which greatly reduces the actual amount of code the implementation requires.

A set of basic function BIC Callout classes will be available as part of the framework, with the expectation that new generic classes will be added as new workflows are implemented. The general-applicability classes will be defined in the Services framework. Project-specific classes will be defined

within the code base for the project. While these classes may not have general applicability, they may have re-use potential within other workflows for the same project (or customer).

Callout BIC Interface

The following methods must be implemented in each CalloutBic class

`String getDescription()`

Returns a detailed description of what the CalloutBic class does. This description coupled with the information for the input and output parameters, it should not be necessary to consult any printed documentation.

`void setInputParameters()`

Defines the input parameters. This method must call `addInputParameter()` for each input parameter.

`void setOutputParameters()`

Defines the output parameters. This method must call `addOutputParameter()` for each output parameter

`boolean processWorkItem(Work work, HashMap inputProperties, HashMap outputParameters, StringBuffer errors)`

Processes a work item. Returns true if the processing was successful. The errors parameter should be populated with any error messages if the processing fails. A `BICException` will be thrown if the processing fails. This will flag the work item with an error (if the activity is not configured to send on error).

- **work** – details on the work item being processed
- **inputProperties** – map of input properties and their values, which have been resolved (if the activity configuration contained work item property references)
- **outputParameters** – map for the output properties and their values generated by the Callout BIC and stored with statements like the following:

```
outputParameters.put(OUTPUT_PARAMETER_NAME,
    valueForParameter);
```

Callout BIC Methods

The following methods are available to each CalloutBic class

`void addInputParameter(String name, String description, String[] options, String codeSetName, String defaultValue, boolean isRequired, int maxValues)`

Adds an input parameter to the list for the CalloutBic

- **name** – name of the parameter
- **description** – detailed description for the parameter

- **options** – optional list of possible values for the parameter or null if none. These will appear in a drop-down selection list for the parameter in the CalloutBic Settings tab. Must be null if codeSetName is defined.
- **codeSetName** – optional name of an Enable code set containing the possible values for the parameter. The referenced values will appear in a drop-down selection list for the parameter in the CalloutBic Settings tab. Must be null if options is defined.
- **defaultValue** – optional default/initial value for the parameter or null if not defined. If options or codeSetName is specified, this value must match a value in the referenced list.
- **isRequired** – set to true if the parameter must be defined
- **maxValues** – specifies the maximum number of values that can be configured for this parameter. This will be set to 1 for most parameters. When set to a value larger than one, the Settings tab for the CalloutBic will show the named parameter with a trailing number, starting at 0. For example, if the parameter is named “Attribute” and the maxValues is set to three, the CalloutBic will show the properties “Attribute0”, “Attribute1”, and “Attribute2”

void addOutputParameter(String name, String description)

Adds an output parameter to the list for the CalloutBic

- **name** – name of the parameter
- **description** – detailed description for the parameter

void cleanup()

Closes all input streams, output streams and releases queries obtained using the newInput(), newOutput(), getQuery(), and epXGetQuery() methods respectively

void closeInput(BufferedReader br)

Closes a BufferedReader that was previously opened by newInput(). It is not necessary to close each opened file if the cleanup() method is called.

void closeOutput(PrintWriter pw)

Closes a PrintWriter that was previously opened by newOutput(). It is not necessary to close each opened file if the cleanup() method is called.

boolean doesFileExist(String directoryName, String filename)

Returns true if the specified file exists in the specified directory.

DBQuery epXGetQuery()

Retrieves a DBQuery instance from the DBQueryPool which can be used for performing queries against the EPX database. The DBQuery instance represents an existing connection to the database. If an exception is encountered with any DBQuery or DBResultSet methods, the close() method on the DBQuery instance should be called to reset the connection. The same connection can be used for queries that are executed sequentially.

NOTE: Any query object that is retrieved using this method needs to be relinquished with the epXFreeQuery() method.

void epxFreeQuery(DBQuery dbQuery)

Releases a DBQuery instance previously obtained with epxFreeQuery() back to the query pool. It is not necessary to close each opened file if the cleanup() method is called.

void fatalError(String message)

Forces the CalloutBic to throw a BICException which will halt the processing of the work item (until its error is cleared)

- **message** – message to logged with fatal error

void freeQuery(DBQuery dbQuery)

Releases a DBQuery instance previously obtained with getQuery() back to the query pool. It is not necessary to close each opened file if the cleanup() method is called.

String getParameter(HashMap inputProperties, String name, String defaultValue)

Returns the resolved value of the specified input parameter.

- **inputProperties** – HashMap passed into the processWorkItem() method
- **name** – name of the input parameter. Any property references in the configured input parameter value are resolved before the value is returned by this method. The name must match an input parameter defined with the addInputParameter() method. If the maxValues for the parameter is greater than 1, the name must identify which parameter (e.g., "Attribute0", "Attribute1", etc.)
- **defaultValue** - returned if the value of the resolved work item property is empty or not defined.

String getSystemProperty(String propertyName)

Returns the value of the specified system property from the work item. Only the following properties can be specified using the properties defined on the BaseCalloutBic class:

- SYS_SYSTEM_SERVER_NAME = "sys.system.servername";
- SYS_SERVER_UID = "sys.server.uid";
- SYS_MAIN_FLOW_NAME = "sys.mainflow.name";
- SYS_WORKFLOW_NAME = "sys.workflow.name";
- SYS_ACTIVITY_NAME = "sys.activity.name";
- SYS_WORK_ITEM_NAME = "sys.workitem.name";
- SYS_WORK_ITEM_DESCRIPTION = "sys.workitem.description";
- SYS_WORK_ITEM_DUE_DATE = "sys.workitem.duedate";
- SYS_WORK_ITEM_INITIATOR = "sys.workitem.initiator";
- SYS_WORK_ITEM_CREATION_DATE = "sys.workitem.creationdate";
- SYS_WORK_ITEM_UID = "sys.workitem.uid";

- `SYS_WORK_ITEM_VERSION_UID = "sys.workitem.version.uid";`
- `SYS_WORK_ITEM_VERSION_STATUS = "sys.workitem.version.status";`
- `SYS_WORK_ITEM_VERSION_ERROR = "sys.workitem.version.error";`
- `SYS_WORK_ITEM_EXPIRED = "sys.workitem.expired";`
- `SYS_WORK_ITEM_VERSION_PENDING_TAG = "sys.workitem.version.pending.tag";`
- `SYS_WORK_ITEM_VERSION_WAIT_DATETIME = "sys.workitem.version.wait.datetime";`

DBQuery getQuery()

Retrieves a DBQuery instance from the DBQueryPool which can be used for performing queries against the Enable (EPIM) database. The DBQuery instance represents an existing connection to the database. If an exception is encountered with any DBQuery or DBResultSet methods, the close() method on the DBQuery instance should be called to reset the connection. The same connection can be used for queries that are executed sequentially.

NOTE: Any query object that is retrieved using this method needs to be relinquished with the freeQuery() method.

void logDebug(String channel, String message)

Logs the specified message on the specified channel if debug logging is enabled for that channel.

- **channel** – name of the debug channel (once implemented). The debug message will only be output if the logging for this channel has been enabled.
- **message** – message to be logged. The message will be prepended with the date and time as well as the name of the CalloutBic and the method from which the call is made

void logError(String message)

Logs an error message

- **message** – message to be logged. The message will be prepended with the date and time as well as the name of the CalloutBic and the method from which the call is made

BufferedReader newInput(String directoryName, String fileName, String charset)

Returns a BufferedReader instance for the specified file in the specified directory and the specified file encoding. The BufferedReader object is tracked by the CalloutBIC framework and will be properly closed by the cleanup() method. Returns null if the file could not be opened.

- **directoryName** – fully-qualified path for the directory containing the file
- **filename** – name of the file to be opened
- **charset** – name of the encoding for the file (e.g., "UTF-8")

PrintWriter newOutput(String directoryName, String fileName, String charset)

Returns a PrintWriter instance for the specified file in the specified directory using the specified file encoding. The PrintWriter object is tracked by the CalloutBIC framework and will be properly closed by the cleanup() method. Returns null if the file could not be created.

- **directoryName** – fully-qualified path for the directory containing the file
- **filename** – name of the file to be opened
- **charset** – name of the encoding for the file (e.g., “UTF-8”)

Example Callout Class

The following example is the entirety of the code for the Send Email Callout BIC.

```
package com.enterworks.services.calloutbic;

import java.util.ArrayList;
import java.util.HashMap;
import com.enterworks.apps.agents.bic.Work;
import com.enterworks.services.common.EmailUtil;

public class SendEmail extends BaseCalloutBic implements
CustomCalloutBic {
    private static final String EMAIL_ATTACHMENT =
"email_attachment";
    private static final String EMAIL_BODY = "email_body";
    private static final String EMAIL_SEND_MESSAGE =
"email_send_message";
    private static final String EMAIL_SEND_STATUS =
"email_send_status";
    private static final String EMAIL_SUBJECT =
"email_subject";
    private static final String EMAIL_FROM = "email_from";
    private static final String EMAIL_TO = "email_to";
    private static final String EMAIL_CC = "email_cc";
    private static final String CHANNEL_SEND_EMAIL_BIC = "Send
Email BIC";

    @Override
    public String getDescription() {

        return "Send an e-mail message to one or more recipients,
identified by any combination of " +
            "e-mail addresses, Enable user logins, or
Enable Groups. The e-mail is queued for " +
```



```

        "processing based on the Email Manager
settings.";
    }

    @Override
    public void setInputParameters() {
        this.addInputParameter(EMAIL_FROM, "Optional from address.
        If empty, global default will be used.", null, null, null,
        false, 1, null);
        this.addInputParameter(EMAIL_TO, "Comma-delimited
        list of any combination of e-mail addresses, " +
        "Enable user logins, or Enable Group
names.", null, null, null, true, 1, null);
        this.addInputParameter(EMAIL_CC, "Comma-delimited
        list of optional CC e-mail address. If empty, global
        default will be used.", null, null, null, false, 1, null);
        this.addInputParameter(EMAIL_SUBJECT, "Subject
        line for e-mail", null, null, null, true, 1, null);
        this.addInputParameter(EMAIL_BODY, "Body text for
        e-mail", null, null, null, true, 1, null);
        this.addInputParameter(EMAIL_ATTACHMENT, "Fully-
        qualified path to file to be attached to the e-mail
        message", null, null, null, false, 1, null);
    }

    @Override
    public void setOutputParameters() {
        this.addOutputParameter(EMAIL_SEND_STATUS,
        "Results of e-mail send operation (SUCCESS or FAIL)");
        this.addOutputParameter(EMAIL_SEND_MESSAGE,
        "Detailed error message if email.send.status is FAIL");
    }

    @Override
    public boolean processWorkItem(Work work, HashMap
    inputProperties, HashMap outputParameters, StringBuffer
    errors) {
        // Retrive properties
        this.logDebug(CHANNEL_SEND_EMAIL_BIC, "Inside
        processWorkItem.");
        String emailFrom =
        this.getParameter(inputProperties, EMAIL_FROM, null);
        String emailTo =
        this.getParameter(inputProperties, EMAIL_TO, null);
    }

```

```

        String emailCc =
this.getParameter(inputProperties, EMAIL_CC, null);
        String emailSubject =
this.getParameter(inputProperties, EMAIL_SUBJECT, null);
        String emailBody =
this.getParameter(inputProperties, EMAIL_BODY, null);
        String emailAttachment =
this.getParameter(inputProperties, EMAIL_ATTACHMENT, null);

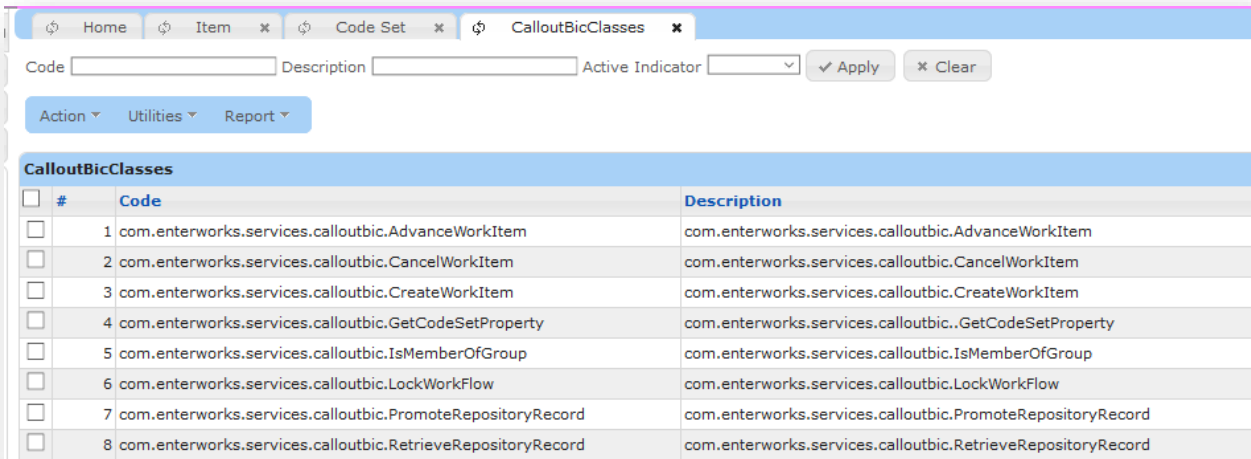
        logDebug(CHANNEL_SEND_EMAIL_BIC, "emailFrom=" +
emailFrom + " emailTo=" + emailTo + " emailCc=" + emailCc
+
        " emailSubject=" + emailSubject + "
emailBody=" + emailBody + " emailAttachment=" +
emailAttachment);
        if (EmailUtil.saveEmail(emailFrom, emailTo,
emailCc, emailSubject, emailBody, emailAttachment)) {
            // Email successfully sent
            outputParameters.put( EMAIL_SEND_STATUS,
"SUCCESS");
        } else {
            outputParameters.put(EMAIL_SEND_STATUS,
"FAIL");
            outputParameters.put(EMAIL_SEND_MESSAGE,
"Failed to send e-mail");
        }
        return true;
    }

}

```

Registering and Configuring a Callout BIC

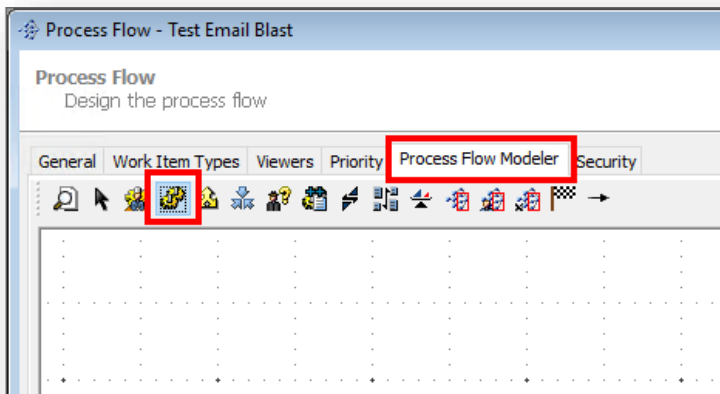
Once a Callout BIC class has been implemented, it needs to be registered in the Enable CalloutBicClasses code set so it can be selected from within Design Console. The fully-qualified classpath must be specified in both the code and description fields in the code set:



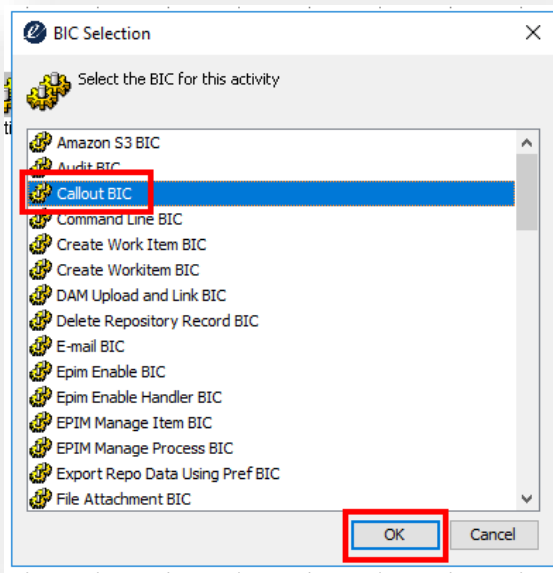
#	Code	Description
1	com.enterworks.services.calloutbic.AdvanceWorkItem	com.enterworks.services.calloutbic.AdvanceWorkItem
2	com.enterworks.services.calloutbic.CancelWorkItem	com.enterworks.services.calloutbic.CancelWorkItem
3	com.enterworks.services.calloutbic.CreateWorkItem	com.enterworks.services.calloutbic.CreateWorkItem
4	com.enterworks.services.calloutbic.GetCodeSetProperty	com.enterworks.services.calloutbic..GetCodeSetProperty
5	com.enterworks.services.calloutbic.IsMemberOfGroup	com.enterworks.services.calloutbic.IsMemberOfGroup
6	com.enterworks.services.calloutbic.LockWorkFlow	com.enterworks.services.calloutbic.LockWorkFlow
7	com.enterworks.services.calloutbic.PromoteRepositoryRecord	com.enterworks.services.calloutbic.PromoteRepositoryRecord
8	com.enterworks.services.calloutbic.RetrieveRepositoryRecord	com.enterworks.services.calloutbic.RetrieveRepositoryRecord

To configure the registered Callout BIC, perform the following steps:

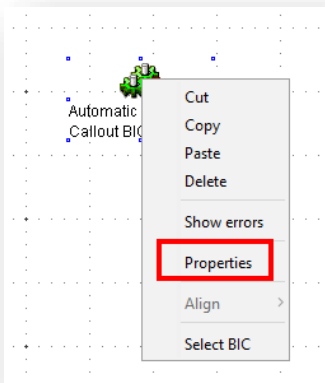
1. Open a Workflow or Subflow in the Design Console.
2. Click on the Process Flow Monitor tab
3. Click on the automatic activity icon:



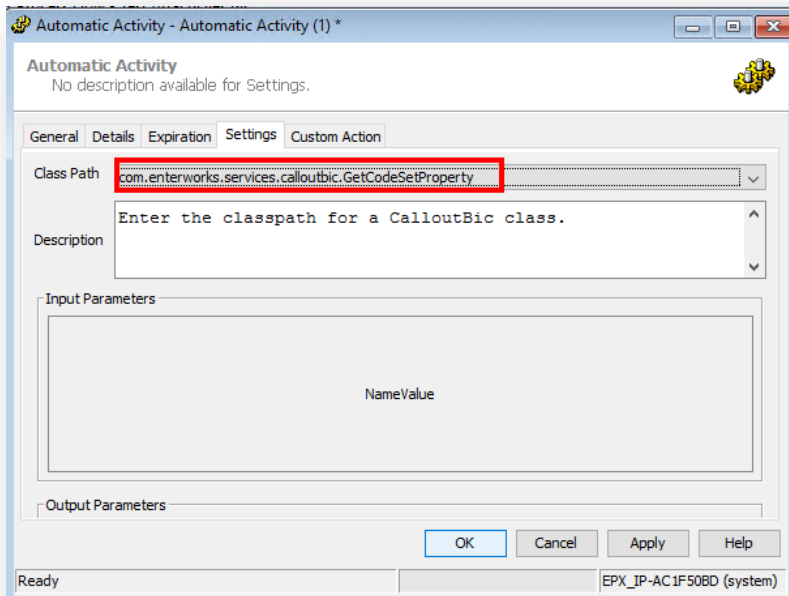
4. Click on the canvas. The BIC Selection prompt appears.
5. Select Callout BIC and click OK:



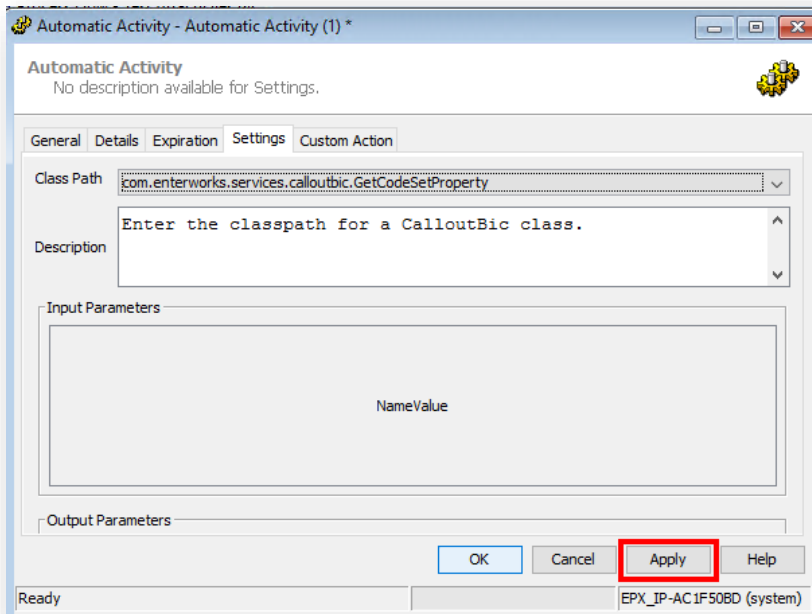
6. Right-click on the green double-gear icon and select Properties from the pop-up menu:



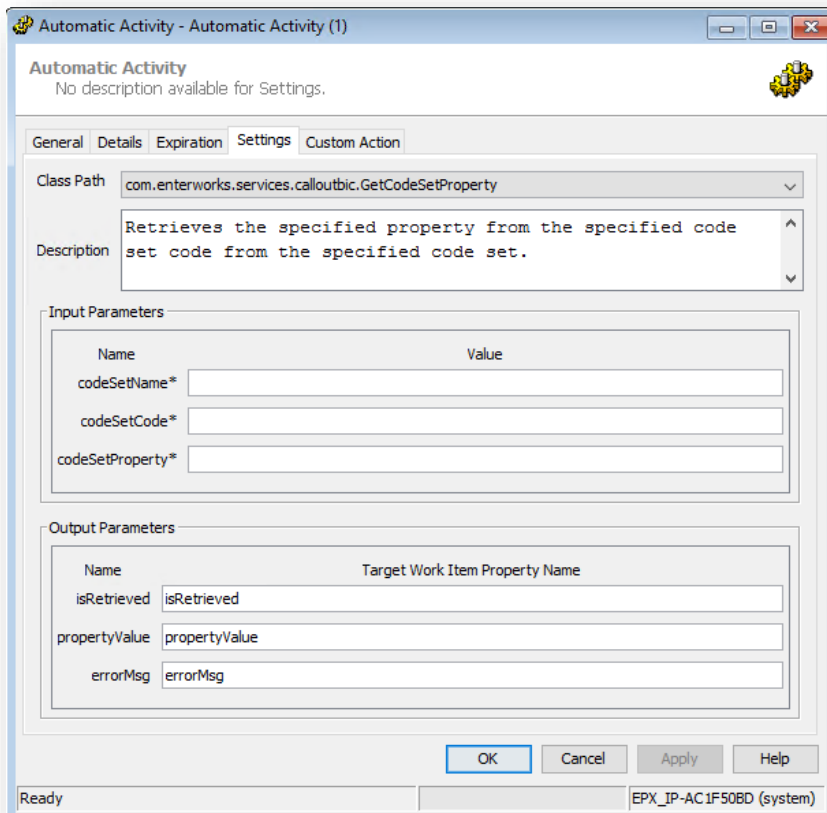
7. Select the desired Callout Class:



8. Click Apply:



9. Configure the class as desired:



Creating Standalone Callout BIC

When a new Callout BIC class is created and registered, it shares the same thread pool as all other Callout BIC classes that are only registered. The Design Console will also show the double-green gear icon for each of the different registered classes. To provide the Callout BIC with its own thread pool and/or to provide a unique icon for the activity in Design Console, a Standalone Callout BIC for the class needs to be created by performing the following steps:

1. Copy the Standalone Callout BIC Template folder from the Services Framework deployment to a temporary directory.
2. Open a PowerShell window.
3. Change the current directory to the copied Standalone Callout BIC Template folder.
4. Enter the command:
`.\GenerateBIC.ps1`
 For example:

```

Windows PowerShell
PS C:\temp\services\Standalone Callout BIC Template> dir

Directory: C:\temp\services\Standalone Callout BIC Template

Mode                LastWriteTime         Length Name
----                -
d-----          1/14/2021  11:45 PM                BICTemplate
-a-----          9/7/2017    9:58 PM             932 BIC Starting Point Flag.gif
-a-----          7/25/2017    3:48 PM             922 BICErrorFlag.gif
-a-----         10/31/2020    4:48 PM            8913 build.xml
-a-----          7/24/2017    2:08 PM            1120 ExampleFlowMonitorInvalid.gif
-a-----          11/1/2020    1:38 AM            5520 GenerateBIC.ps1

PS C:\temp\services\Standalone Callout BIC Template> .\GenerateBIC.ps1

```

5. Enter the name of the BIC as it is to appear in the Design Console:

```

Windows PowerShell
Services Framework Callout BIC Generation

This script generates a new Callout BIC
BIC Label (e.g., Send Email)? Read Rabbit MQ BIC

```

6. Enter the name of the package for the BIC:

```

Windows PowerShell
Services Framework Callout BIC Generation

This script generates a new Callout BIC
BIC Label (e.g., Send Email)? Read Rabbit MQ BIC
Name of Java Package for BIC (e.g., com.enterworks.services.epx.bic.sendemailbic)? com.enterworks.bic.readrabbitmqbic

```

7. Enter the name for the BIC Class. This class can be defined in a separate JAR file or it can be added to the Standalone Callout BIC package so the BIC is self-contained. All of the Services

Framework Standalone Callout BICs reside in the Services.jar file. This allows the processing code for the Callout BICs to be deployed/updated in a single file but also sets up a dependency to the Services.jar file:

```
Windows PowerShell
Services Framework Callout BIC Generation

This script generates a new Callout BIC
BIC Label (e.g., Send Email)? : Read Rabbit MQ BIC
Name of Java Package for BIC (e.g., com.enterworks.services.epx.bic.sendemailbic)? : com.enterworks.bic.readrabbitmqbic
Callout Classpath (e.g., com.enterworks.services.calloutbic.SendEmail)? : com.enterworks.calloutbic.ReadRabbitMqBic
```

8. Enter the source directory path where the files for the BIC class will be placed. This should include all of the package fields except the last one as the final part of the path:

```
Windows PowerShell
Services Framework Callout BIC Generation

This script generates a new Callout BIC
BIC Label (e.g., Send Email)? : Read Rabbit MQ BIC
Name of Java Package for BIC (e.g., com.enterworks.services.epx.bic.sendemailbic)? : com.enterworks.bic.readrabbitmqbic
Callout Classpath (e.g., com.enterworks.services.calloutbic.SendEmail)? : com.enterworks.calloutbic.ReadRabbitMqBic
Filepath for new BIC source folder (e.g., C:\Development\Enterworks\BICs\)? : C:\Temp\Services\src\com\enterworks\bic
```

9. Review the settings. If they are correct, enter “y”:

```
Windows PowerShell
Services Framework Callout BIC Generation

This script generates a new Callout BIC
BIC Label (e.g., Send Email)? : Read Rabbit MQ BIC
Name of Java Package for BIC (e.g., com.enterworks.services.epx.bic.sendemailbic)? : com.enterworks.bic.readrabbitmqbic
Callout Classpath (e.g., com.enterworks.services.calloutbic.SendEmail)? : com.enterworks.calloutbic.ReadRabbitMqBic
Filepath for new BIC source folder (e.g., C:\Development\Enterworks\BICs\)? : C:\Temp\Services\src\com\enterworks\bic

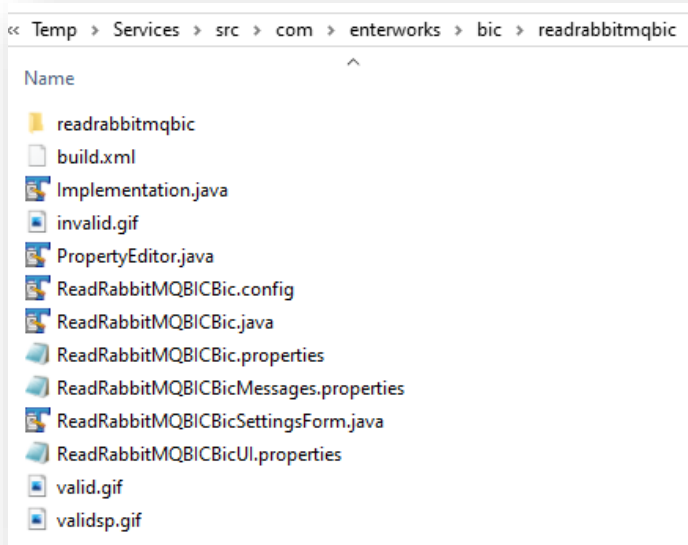
Confirm settings:
Creation Date = 1/15/2021 2:22:38 AM
BIC Label = Read Rabbit MQ BIC
Name of BIC = ReadRabbitMQBIC
Name of Java Package for BIC = com.enterworks.bic.readrabbitmqbic
Callout Classpath = com.enterworks.calloutbic.ReadRabbitMqBic
Source Path For BIC = com/enterworks/bic/readrabbitmqbic
Filepath for BIC = C:\Temp\Services\src\com\enterworks\bic
BIC Root Path = C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic
Continue using these settings? Default [n]: y
```

10. The script will create the directories, copy the files, and update them to reflect the settings entered:

EnterWorks Workflow Activities and Callout BIC Framework

```
Windows PowerShell
Read Rabbit MQ BIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicSettingsForm.java @BIC_NAME@
ReadRabbitMQBIC C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicSettingsForm.java @BIC_PACKAGE_NAME@
com.enterworks.bic.readrabbitmqbic C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicSettingsForm.java @BIC_CALLOUT@
com.enterworks.calloutbic.ReadRabbitMQBic C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicSettingsForm.java @BIC_SOURCE_PATH@
om/enterworks/bic/readrabbitmqbic C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicSettingsForm.java @BIC_DATE@
1/15/2021 02:22:38
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_DATE@ 01/15/2021 02:22:38
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_LABEL@ Read Rabbit MQ BIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_NAME@ ReadRabbitMQBIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_PACKAGE_NAME@ com.enterworks.bic.readrabbit
om.enterworks.bic.readrabbitmqbic C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_CALLOUT@ com.enterworks.calloutbic.Re
bitMQBic C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_SOURCE_PATH@
om/enterworks/bic/readrabbitmqbic C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\ReadRabbitMQBICBicUI.properties @BIC_DATE@ 01/15/2021 02:22:38
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_DATE@ 01/15/2021 02:22:38
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_LABEL@ Read Rabbit MQ BIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_NAME@ ReadRabbitMQBIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_PACKAGE_NAME@ com.enterworks.bic.readrabbit
c C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_CALLOUT@ com.enterworks.calloutbic.ReadRabbit
c C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_SOURCE_PATH@ com/enterworks/bic/readrabbit
c C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\Implementation.java @BIC_DATE@ 01/15/2021 02:22:38
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_DATE@ 01/15/2021 02:22:38
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_LABEL@ Read Rabbit MQ BIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_NAME@ ReadRabbitMQBIC
C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_PACKAGE_NAME@ com.enterworks.bic.readrabbit
c C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_CALLOUT@ com.enterworks.calloutbic.ReadRabbit
c C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_SOURCE_PATH@ com/enterworks/bic/readrabbit
c C:\Temp\Services\src\com\enterworks\bic\readrabbitmqbic\PropertyEditor.java @BIC_DATE@ 01/15/2021 02:22:38
PS C:\temp\services\Standalone Callout BIC Template>
```

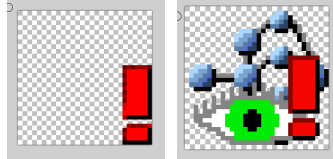
11. Review the files that were generated:



Note: If the Callout BIC class is to be contained within the BIC JAR file, it will need to be copied to this directory.

12. Edit the *.gif files to contain the desired images which must be 64x64 pixels in size. There are three images that need to be updated:

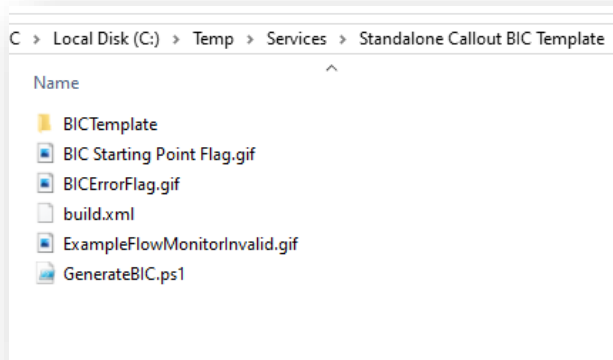
- a. valid.gif – image that is displayed when the activity is valid
- b. invalid.gif – image that is displayed when the activity has a misconfiguration. This should include the BICErrorFlag.gif image on top of the valid.gif image:



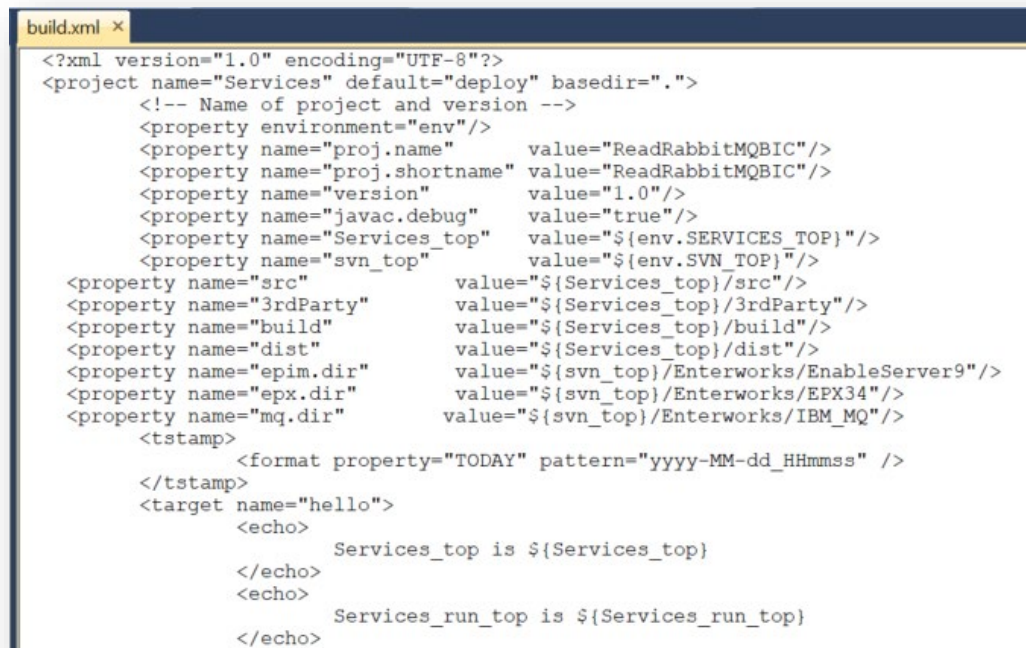
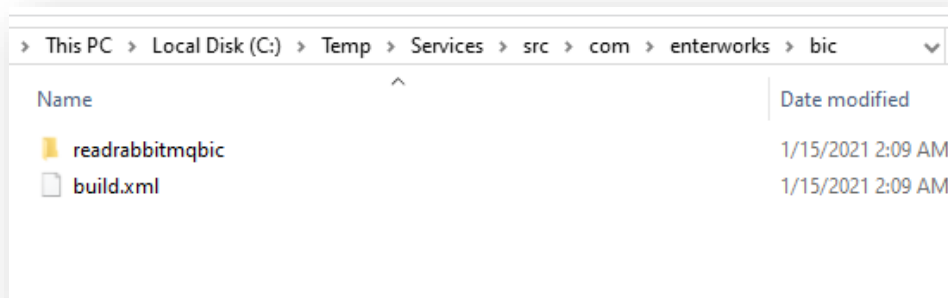
- c. validsp.gif – image that is displayed when the activity is a starting point. This should include the BIC Starting Point Flag.gif image on top of the valid.gif image:



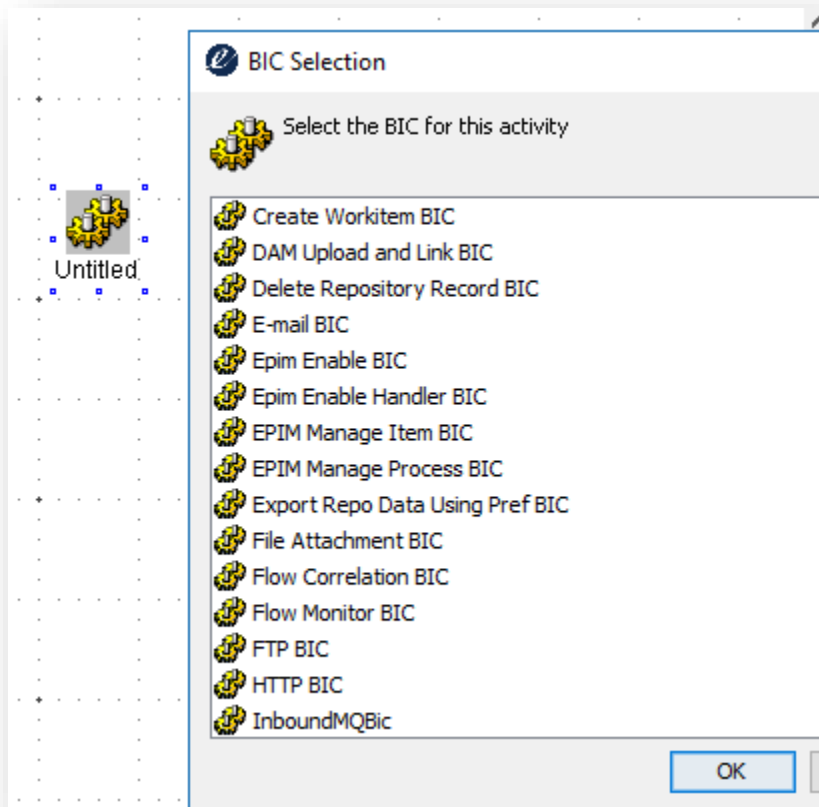
- d. TIP: Use a paint program that supports layers (e.g., paint.net) so the error flag or starting point flag can be added on top of the normal image, then saved as a flattened GIF image.
- e. The Callout BIC Template folder includes the error and starting point images:



13. Integrate the build.xml file into your existing source development. Or simply export a JAR file from your IDE for the BIC :



14. The build.xml script expects the following environment variables to be defined (these can be renamed in the script):
 - a. SERVICES_TOP – root folder of the source tree containing the new BIC
 - b. SVN_TOP – root folder containing the Enterworks project, which contains JDKs, Enterworks Product and Services Framework libraries, ANT and other utilities
15. When the Standalone Callout BIC build is run, a JAR file and config file will be placed in the dist\lib folder under the SERVICES_TOP folder. These files will need to be copied to the <drive>:\Enterworks\EPX\bin\agents\<server> folder on the EPX server to deploy the BIC. If the Callout BIC class is in a separate JAR file, it will need to be deployed as well (and identified in the EPX_Web_Wrapper.conf and DesignConsole.lax files under the <drive>:\Enterworks\EPX\bin folder). Once the files have been deployed, EPX Services must be restarted. The Design Console will show the label for the Standalone Callout BIC in the automatic activity selection list:



Callout BIC Modules

The Callout BIC-based BICs have been implemented for common functions and are detailed in the subsections below. Each BIC can be selected in the automatic activity selection list in Design Console. A graphical icon will represent what the callout does vs. the green double-gears that is displayed for the Callout BIC. Each individual Callout BICs has its own thread pool (the default is 100 maximum threads), so a high-volume of work items at one BIC of a specific type won't prevent any new work items at a different BIC from being processed.

Advance Work Item BIC



Advances a work item on a workflow.

Inputs

- workItemId - ID of the work item to be advanced
- workflowName - Name of the main workflow in which the work item resides
- activityName - Name of the activity in which the work item should be found

- workItemName - Name of the work item to be advanced
- wipName - Name of a work item property to be set
- wipValue - Value of a work item property to be set

Outputs

isAdvanced - Work item was successfully advanced if true

errorMsg - Details for failure if work item was not advanced

Amazon S3 BIC



Transfers one or more files between Amazon S3 and a local directory. Files can be uploaded to or downloaded from Amazon S3. Wildcards can be specified in the file name to find files matching a pattern. Multiple files can be transferred (up to a specified maximum), providing the files match the specified wildcard pattern. A delay can be specified after each file transfer to give the target system a chance to process the file. For example, if this callout is being used to download files directly into the Enable DAMDrop folder, having a delay between files will give Enable time to process each file and upload it to S3 to avoid needing a large amount of file storage space if there are a large number of files needing to be transferred.

Inputs

- fileTransferDirection - Direction of file transfer."
- awsEndpoint – Amazon S3 Endpoint name. Not set if blank (default). Use configuration repository property named **AWS.S3.Endpoint** if set to 'Use Config'
- awsRegionName – Amazon S3 Region Name. Not set if blank (default). Required if awsEndpoint specified. Use configuration repository property named **AWS.S3.Region** if set to 'Use Config'
- awsBucketName - Amazon S3 Bucket Name.
- cannedAccessControlList - Specifies the Canned Access Control List. Used default if blank
- sourceDirectoryName - Name of directory in Amazon S3 Bucket (download) or local file system (upload) containing file(s) to be transferred.
- sourceFileName - Name of file in Amazon S3 (download) or local file system (upload) to be transferred. If the name includes one or more wildcard characters ('*'), the first maxFilesToTransfer matching file(s) found will be retrieved.
- targetDirectoryName - Target directory in local file system (download) or Amazon S3 (upload) where files are to be transferred.
- maxFilesToTransfer - Maximum number of files to transfer if the source file name has a wildcard. 'All' will transfer all matching files (up to 1000).

- fileTransferDelay - Number of milliseconds to delay after transferring each file. This can be used to meter downloads of large numbers of files destined for the DAMDrop folder to give Enable DAM time to process the files to avoid a significant surge in file storage needed.
- deleteFileAfterTransfer - Deletes the file from the source after successful transfer to the target if true.
- isRoleBasedAuth - Use Role-based authentication if true, otherwise use secret key
- awsAccessKey - AWS Access key if isRoleBasedAuth is false
- awsSecretKey - AWS Secret key if isRoleBasedAuth is false

Outputs

- transferStatus - Results of AWS transfer operation (SUCCESS or FAIL)
- filesFound - Number of files found for transfer
- filesTransferred - Number of files successfully transferred
- errorMessage - Detailed error message if transferStatus is FAIL

The following screenshot is an example of the Amazon S3 BIC configured to upload files to S3:

Automatic Activity - Upload to S3

Automatic Activity
No description available for Settings.

General Details Expiration Settings Custom Action

Class Path: com.enterworks.services.calloutbic.AmazonS3Callout

Description: Transfers one or more files between Amazon S3 and a local directory. Files can be uploaded to or downloaded from Amazon S3. Wildcards can be specified in the file name to find files matching a pattern. Multiple files can be transferred (up to a specified)

Input Parameters

Name	Value
fileTransferDirection*	Upload
awsBucketName*	services.enterworks.com
sourceDirectoryName	D:/temp/Uploads
sourceFileName*	*.txt
targetDirectoryName	Downloads
maxFilesToTransfer*	3
fileTransferDelay*	5000
deleteFileAfterTransfer*	true
isRoleBasedAuth*	false

Output Parameters

Name	Target Work Item Property Name
transferStatus	transferStatus
filesFound	filesFound
filesTransferred	filesTransferred
errorMessage	errorMessage

OK Cancel Apply Help

Ready EPX_EWAPP1 (system)

In the above example, up to 3 files with the extension “.txt” in the D:\temp\Uploads folder will be placed in the Downloads folder under the services.enterworks.com bucket in S3 with a 1/2 second delay after each file has been uploaded. The files are deleted from D:\temp\Uploads if they have been uploaded successfully.

The following screenshot is an example of the Amazon S3 BIC configured for download:

Automatic Activity - Download From S3

Automatic Activity
No description available for Settings.

General Details Expiration Settings Custom Action

Class Path: `com.enterworks.services.calloutbic.AmazonS3Callout`

Description: Transfers one or more files between Amazon S3 and a local directory. Files can be uploaded to or downloaded from Amazon S3. Wildcards can be specified in the file name to find files matching a pattern. Multiple files can be transferred (up to a specified

Input Parameters

Name	Value
fileTransferDirection*	Download
awsBucketName*	services.enterworks.com
sourceDirectoryName	Downloads
sourceFileName*	*.rpt
targetDirectoryName	D:/temp/Downloads
maxFilesToTransfer*	3
fileTransferDelay*	5000
deleteFileAfterTransfer*	true
isRoleBasedAuth*	false

Output Parameters

Name	Target Work Item Property Name
transferStatus	transferStatus
filesFound	filesFound
filesTransferred	filesTransferred
errorMessage	errorMessage

OK Cancel Apply Help

Ready EPX_EWAPP1 (system)

In the above example, up to 3 files with the extension “.rpt” in the Downloads folder under the services.enterworks.com bucket in S3 will be placed in the D:\temp\Downloads with a 1/2 second delay after each file has been downloaded. The files are deleted from Amazon S3 if they have been downloaded successfully.

Attach Files BIC



Attaches files to the work item. Can either specify a newline or comma-delimited list of names. If newline is used as the delimiter, file names do not have to be escaped if they include commas. If comma is the delimiter, any file name containing a comma must be surrounded by double quotes.

Inputs

- `directoryPath` - Optional base directory for files. If defined, it will be prefixed to each listed file.
- `fileNames` - Newline-delimited or comma-delimited list of files. Only one delimiter can be used. If a newline character is found, it will be used as the delimiter. If a file name contains a comma, the file name must be surrounded by double-quotes unless newline is the delimiter. This includes when there is only one file being attached.
- `fileName0-19` - Attach a single file. File names with commas do NOT have to be surrounded by double quotes.

Outputs

- `attachStatus` - Attach files operation completely successful if SUCCESS. Any failure will return FAIL.
- `messages` - Details for processing of the files
- `filesListed` - Number of files listed for attachment (reflects files found when wildcard is used in file name).
- `filesAttached` - Number of files successfully attached.

CreateWorkItem BIC



Creates a new work item on the specified workflow

Inputs

- `userLogin` - Login name of the user under which to create the work item (e.g., `%initiatedUser%`)
- `workflowName` - name of the workflow on which to create a work item
- `activityName` - name of the starting point activity at which to create the work item
- `workItemName` – name of the work item (if settable for the workflow)
- `workItemDescription` - optional description for the work item
- `wipName1-20` - the names of up to 20 work item properties to be created
- `wipValue1-20` - the values for the work item properties. Empty values will be defined as empty in the new work item

Outputs

- `isCreated` - work item was successfully created if true
- `errorMsg` - detailed message of why the work item was not created

DAM Upload and Link BIC



Uploads the designated file to the DAM and links it to the designated record in the designated repository.

Inputs

- repositoryName - Name of repository in in which a record is to be updated/created
- id - Internal ID of the repository record to be updated (0 or blank if none)
- filePath - Fully-qualified path of the file to be uploaded
- imageCaption - Optional caption for linked file
- imageContext - Context for linked image.
- imagePriority - Optional priority for the image.

Outputs

- uploadAndLinkStatus - Status of upload and link operation. SUCCESS if file is uploaded and linked successfully.
- errorMsg - Detailed message why the file was not uploaded and linked if status FAILED

Delete Repository Record BIC



Deletes a repository record identified by the ID or designated primary key (which does not have to be the repository primary key).

Inputs

- repositoryName - Name of repository in in which a record is to be deleted.
- id - Internal ID of the repository record to be deleted (0 or blank if none)
- pkName(TBD) - Attribute name for primary key (actual or alternative) of record to be deleted.
- pkValue(TBD) - Attribute value for primary key (actual or alternative) of record to be deleted.
- enableTrigger(TBD) - Activate trigger on target repository if true

Outputs

- isDeleted - Set to true if the designated record(s) are deleted.
- errorMsg - Detailed message why the record could not be created or updated (if ID is '0')

Export Repo Data Using Pref BIC



Generate an export file from a specified source repository to a using the designated User Preference

Inputs

- source_repo_name - The source repository name
- source_preference_name - The source repository preference name
- source_savedset_id - The savedSetId of the source repository
- source_item_ids - If no savedSetId is specified, a Comma-delimited list of item ids can be specified
- targetDirectoryName - Fully qualified path to directory where export file is to be placed.
- targetFileName - Base name for the export file (timestamp and extension will be added to the name).
- targetFileType - Type of file to be generated.
- exportTimeoutSeconds - Number of seconds before timeout declared for export

Outputs

- exportStatus - Results of export operation (SUCCESS or FAIL)
- exportMessage - Detailed error message if exportStatus is FAIL
- exportFilePath - Fully-qualified path for export file if exportStatus is SUCCESS

Flow Monitor BIC



Monitors all or a subset of workflows for work items stuck with errors. Generates an HTML report and clears the errors on the work items so reprocessing is attempted. The system_check.jsp page under EPX needs to be updated to include the check and reporting of the workflow errors file.

Inputs

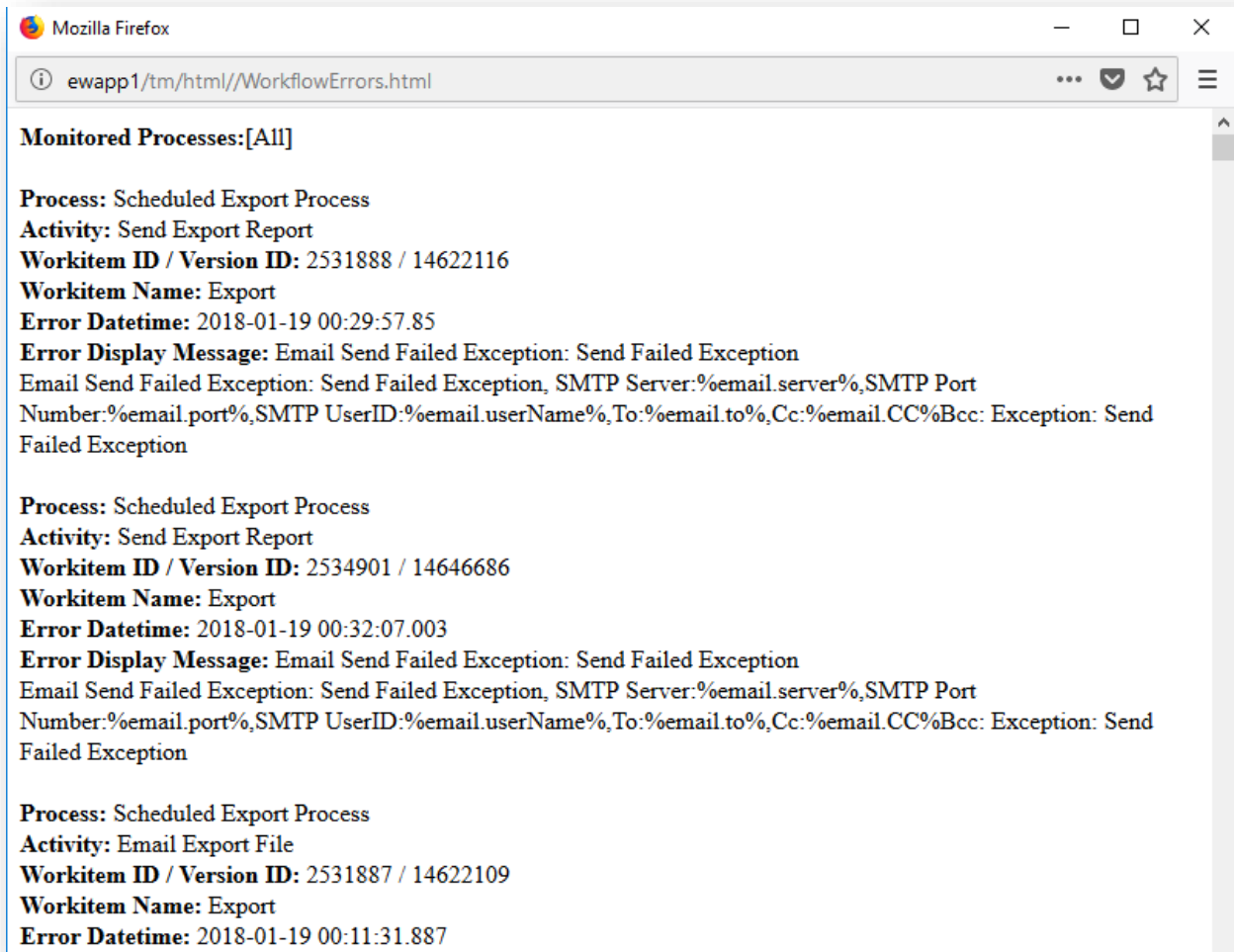
- User Name – Login of an EPX user with Administrator group assigned
- User Password – Password for the EPX user
- Results Key – work item property prefix for all properties generated by the BIC
- Monitored Processes – list of workflows to be monitored with the asterisk (*) acting as a wildcard. A value of just the asterisk will monitor all workflows in EPX

Outputs

- Generates an HTML report that is accessible from the System Health Widget:

System Health		
SERVICE	STATUS	MESSAGE
EPIM_TOMCAT_MASTER	✓	OK
EPIM_DATABASE	✓	OK
EPIM_JBOSS_MASTER	✓	OK
EPIM_JBOSS_SLAVE	✓	no slaves installed.
EPIM_JOB_MONITOR	⚠	0 running jobs. 3 queued jobs. 2 jobs errored in last day.
EPIM_TOMCAT_FILE_SYSTEM	✓	OK
EPIM_JBOSS_MASTER_FILE_SYSTEM	✓	OK
EPX_TOMCAT	✓	OK
EPX_JBOSS	✓	OK
EPX_BIC_MANGER	✓	OK
EPX_WORKFLOW_STATUS	⚠	Workflow Errors.
EPX_FILE_SYSTEM	✓	OK

Clicking the Workflow Errors link opens the report, listing each work item that is flagged with an error:



Initiate Scheduled Export BIC



Initiate a scheduled export by overlaying the listed properties.

Inputs

- **scheduledExportName** - Name of scheduled export to be initiated. Must exactly match the Export Name in a Scheduled Export record.
- **keyNameX** - Name of scheduled export property to override.
- **keyValueX** - Value for the corresponding of scheduled export property to override

Outputs

- **action** - Results of export operation (success or fail)

- errMsg - Detailed error message if action is fail

Initiate Scheduled Import BIC



Initiate scheduled imports by sending in an input parameter of comma-delimited list of scheduled imports. Random UUID numbers are generated for each scheduled import and result stored in Parameter1 for each corresponding scheduled import for later scheduled import job status tracking. If the optional name/value pairs are defined, the Parameter1 attribute is NOT populated with the UUID. Instead only the designated attributes are updated with the specified values.

Inputs

- scheduledImports - Comma-delimited list of scheduled imports to initiate.
- importAttributeName_N - Name of the Scheduled Import Attribute to be updated with importAttributeValue_<number>. If the first field is defined, the UUID is not generated and instead only the specified name/value pairs are updated in the designated attributes in the Import job being launched. Where N is from 0 to 19
- importAttributeValue_N - Name of the Scheduled Import Attribute to be updated with importAttributeValue_N

Outputs

- ScheduledImportsParameters - Colon-delimited list of Parameter1s for each corresponding inputs of Scheduled Imports
- action - Results of e-mail send operation (success or fail)
- errMsg - Detailed error message if action is fail

Lock Export Data BIC



Acquires a lock for a Scheduled Export based on the lock settings in that Export. A separate BIC was created to isolate the lock requests from the maximum threads used by some of the activities after the lock (Universal Java BIC activities) to avoid deadlocks. If the lock action is overwhelmed with a large number of work items, The newest work items will be blocked from acquiring lock but the work items that acquire a lock will still be able to process.

Inputs

- exportJob - Export Job Number
- lockTimeoutPeriodSQL - If in wait for lock mode and timeout period expires for sql query, force the lock
- lockRetryInterval - specify the retry interval if in wait for lock mode and the action acquireLock failed
- workItemId - ID of the work item to be lockedOutputs

Outputs

- lockName - Name of the lock (if successfully acquired)
- lockActionStatus - Results of lockAction (SUCCESS or FAIL)
- lockActionErrMsg - Detailed error message if lockAction is FAIL

Lock Import Data BIC



Acquires a lock for a Scheduled Import based on the lock settings in that Import. A separate BIC was created to isolate the lock requests from the maximum threads used by some of the activities after the lock (Universal Java BIC activities) to avoid deadlocks. If the lock action is overwhelmed with a large number of work items, the newest work items will be blocked from acquiring lock but the work items that acquire a lock will still be able to process.

Inputs

- importJob - Import Job Number.
- lockTimeoutPeriodSQL - If in wait for lock mode and timeout period expires for sql query, force the lock.
- waitForLock - Specify if wait for lock and return success status or return immediately with success/fail status
- timeoutPeriodSql - If in wait for lock mode and timeout period expires for sql query, force the lock
- lockRetryInterval- specify the retry interval if in wait for lock mode and the action acquireLock failed
- workItemId - ID of the work item to be locked

Outputs

- lockName - Name of the lock (if successfully acquired)
- lockActionStatus - Results of lockAction (SUCCESS or FAIL)
- errorMessages - Detailed error message if lockAction is FAIL

Lock Workflow BIC



Acquires a lock to ensure only one work item continues in the workflow at a time. Must be followed by an Unlock Workflow activity that specifies the same lock name. This activity can be configured to wait for the lock or return immediately. In either case, the designated output property indicates whether the lock was successfully acquired. Waiting for a lock should only be done when each work item contains work that must continue in the workflow. In cases where the work item is being generated by scheduled (e.g., in cases where the work item is going to check for work and terminate if there isn't any),

the work item should NOT wait for the lock but instead check whether the lock was acquired after this activity and terminate if it was not.

Inputs

- lockName - Name of the lock. This is a global name across Enable, meaning for example, if a Scheduled Export job as set the lock 'MyLock' will cause work items to wait for that lock to be cleared if this is set to 'MyLock' as well.
- lockAction - Specify whether to wait for an exclusive Lock or attach to a Group lock. An exclusive lock means if another process has the same lock, the lock will not be acquired. If a group lock, the work item will be attached to the same lock.
- waitForLock - Specify if wait for lock and return success status or return immediately with success/fail status
- timeoutPeriodSql - If in wait for lock mode and timeout period (in minutes) expires for sql query, force the lock
- lockRetryInterval - specify the retry interval if in wait for lock mode and the action acquireLock failed
- workItemId - ID of the work item to be locked

Outputs

- lockActionStatus - Results of lockAction (SUCCESS or FAIL)
- errorMessages - Detailed error message if lockAction is FAIL

Missing Variants Report BIC



Initiate image report by sending in name of the server containing the JBoss slave process that is targeted for running the utility. Output parameter would be job id for image report utility job.

Inputs

- serverName - Name of the server containing the JBoss slave process that is targeted for running the utility.

Outputs

- imageReportJobId - Job id of initiated image report job
- initiateStatus - Results of initiation of image report operation (SUCCESS or FAIL)
- errorMessages - Detailed error message if action is FAIL

Promote Repository Record BIC



Promotes a single record or a collection of records in a Saved Set from Staging to Production. Records are validated and promoted based on the designated validation mode.

Inputs

- id - Internal ID of record to be promoted
- savedSetId - ID of saved set containing the records to be promoted
- savedSetName - Name of saved set containing the records to be promoted
- promotionConfiguration - Name of the promotion configuration to be used for the promotion
- repositoryName - Name of the staging repository
- validationMode - Validation mode for promotion: All - promote all records (don't validate), Valid - validate and promote only records with no errors, Warning - validate and promote only records with no errors or warnings only.
- promotionTimeout - Specifies the number of minutes the job will wait before declaring the promotion a failure.

Outputs

- isPromoted – true if the promotion operation was successful
- errorMsg – details of failure if the promotion was not successful

Regenerate Variants BIC



Initiate regenerate variant jobs for as many as the scheduled imports defined. Input parameters are: Comma-delimited list of scheduled imports, for which regenerate variant jobs will be initiated, with their saved sets as arguments; Name of the server containing the JBoss slave process that is targeted for running the utility. Output parameter would be a colon-delimited list of job ids for the initiated regenerate variant jobs.

Inputs

- serverName - Name of the server containing the JBoss slave process that is targeted for running the utility.
- scheduledImports - Comma-delimited list of scheduled imports, for which regenerate variant jobs will be initiated, with their saved sets as arguments.

Outputs

- regenerateVariantJobIds - Colon-delimited list of job ids for the initiated regenerate variant jobs
- initiateStatus - Results of initiation of image report operation (SUCCESS or FAIL)
- errorMessages - Detailed error message if action is FAIL

RepoDataCopyBic



Copy data from a specified source repository to a target repository by generating an export file using the designated User Preference and then importing the data into the target repository using the designated import template. This BIC would typically be used when moving data from a request repository to the actual data repository.

Inputs

- source_repo_name - The source repository name "", null, null, null, true, 1, null);
- source_preference_name - The source repository preference name
- source_savedset_id - The savedSetId of the source repository
- source_item_ids - If no savedSetId is specified, a Comma-delimited list of item ids can be specified
- target_repo_name - The target repository name
- target_repo_import_template_name - The import template name of the target repository
- targetFileType - Type of file to be generated.
- keepRepoValues - Retain repository values if value in file is empty if Yes (default is Yes)
- validateAfterCopy - Validate the records after they have been copied if Yes (default is No)

Outputs

- repo_data_copy_status - Results of e-mail send operation (SUCCESS or FAIL)
- repo_data_copy_message - Detailed error message if email.send.status is FAIL

RepoDataEmailBic



Exports data from a repository using a saved set or list of IDs and user preference to control content. The resulting file is sent as an attachment in an e-mail message to one or more recipients, identified by any combination of e-mail addresses, Enable user logins, or Enable Groups. The e-mail is queued for processing based on the Email Manager settings.

Inputs

- email_to - Comma-delimited list of any combination of e-mail addresses, Enable user logins, or Enable Group names.
- email_subject - Subject line for e-mail
- email_body - Body text for e-mail

- repo_name - Used to get epim repository records for email body text
- preference_name - Used to get epim repository records for email body text
- savedset_id - Used to generate an epim repository records export file for email attachment file
- item_ids - Used to get epim repository records for email body text
- export_file_name_prefix - The prefix of the generated export file name
- export_file_type - Select a export file type: xlsx or csv
- email_attachment - Fully-qualified path to optional additional file to be attached to the e-mail message

Outputs

- email_send_status - Results of e-mail send operation (SUCCESS or FAIL)
- email_send_message - Detailed error message if email.send.status is FAIL

RepoDataUpdateBic



Update the repository data by specified repo name, savedSetId or itemIds, and the attribute name-value list. A background Enable job is launched to perform the update.

Inputs

- repo_name - Used to get epim repository records for email body text
- savedset_id - Used to generate an epim repository records export file for email attachment file
- item_ids - Used to get epim repository records for email body text
- update_attr_names - The name list of the attributes to be updated.
- update_attr_values - The value list of the attributes to be updated.

Outputs

- repo_data_update_status - Results of repo data update operation (SUCCESS or FAIL)
- repo_data_update_message - Detailed error message if repo data update is FAIL

RepoTargetDataUpdateBic



Update the target repository data by specifying the source repository name, itemids, using the source repository attributes to specified target repo name, the identify attribute name-value list, and the update attribute name-value list.

Inputs

- srcRepoName - Specify the source repository
- srcRepoItemIds - Specify the source repository itemIds
- srcRepoAttrForTargetRepoName - Specify the source repository attribute used to set the target repository name
- srcRepoAttrs_forTargetRepoidentAttrs - Specify the source repository attributes used to set the target repository identifier attribute names
- srcRepoAttrs_forTargetRepoidentVals - Specify the source repository attributes used to set the target repository identifier attribute values
- srcRepoAttrs_forTargetRepoUpdateAttrs - Specify the source repository attributes used to set the target repository update attribute names
- srcRepoAttrs_forTargetRepoUpdateVals - Specify the source repository attributes used to set the target repository update attribute values.

Outputs

- repo_target_data_update_message - Results of target repo data update operation (SUCCESS or FAIL)
- repo_target_data_update_status - Detailed error message if target repo data update is FAIL

SendEmail Callout



Sends an e-mail to the designated recipient(s) with the specified subject, body, and attachments. Emails sent using this callout (vs. the Email BIC) are subject to the Email Manager configuration (described below)

Inputs

- email_from - Optional from address. If empty, global default will be used.
- email_to - Comma-delimited list of any combination of e-mail addresses, Enable user logins, or Enable Group names.
- email_cc - Comma-delimited list of optional CC e-mail address. If empty, global default will be used.
- email_subject - Subject line for e-mail
- email_body - Body text for e-mail
- email_attachment - Fully-qualified path to file to be attached to the e-mail message

Outputs

- email_send_status - Results of e-mail send operation (SUCCESS or FAIL)
- email_send_message - Detailed error message if email.send.status is FAIL

Send Rabbit MQ BIC



Sends a message on a Rabbit MQ message queue.

Inputs

- queueName – Name of the Rabbit MQ Queue.
- message – Message to be sent.
-

Outputs

- messageSent - Set to true if message was sent successfully
- messageSendStatus - Detailed error message if message was not sent successfully

Set Properties BIC



Sets properties in a work item.

Inputs

- inOutNameX - Name of the work item property to be created/updated with the corresponding value.
- inOutValueX - Value to be stored in the corresponding work item property.

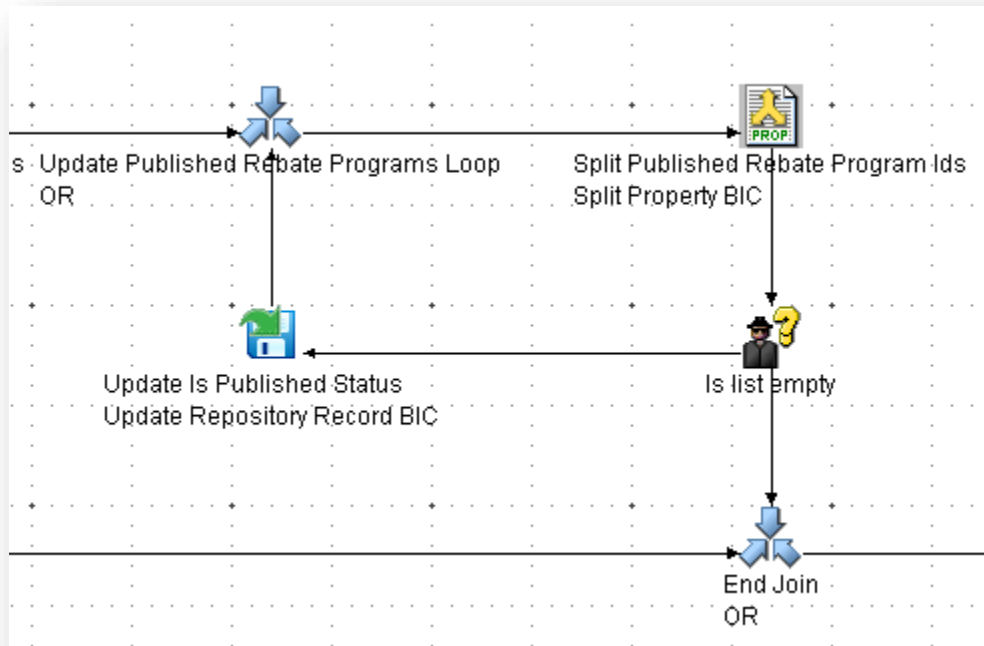
Outputs

- (the named attributes in the inOutNameX input properties.

Split Property BIC



Splits a delimited property value into two values, pulling a single value off from the beginning or end of the delimited list with the remaining value in a second property. This activity can be used within a loop to process one value at a time from a delimited list:



Inputs

- delimitedList - Delimited list of values to be split.
- splitEnd - Specifies whether value is removed from beginning (first) or end (last) of delimited list.
- delimiter - Identifies the delimiter to be used to separate the values in the delimited list.

Outputs

- singleValue - Single value pulled from the delimited list. Empty if list is empty.
- remainingList - Remaining delimited list after pulling off one value. Empty if no values remain.
- isEmpty - Set to true if the delimited list is empty, otherwise false

Split Work Item BIC



Splits a work item with multiple item references to separate work items with the same properties. This should be used for workflows that only support single work items but may have been initiated with either multiple records selected or with a saved set selected.

Inputs

- userLogin - Login name of the user under which to create the split work items (e.g., %initiatedUser%)

- repositoryName - Name of repository containing items
- id - Comma-delimited list of internal record IDs of the records in workflow that need to be split into separate work items. If only a single ID is present, then this activity does nothing and sends the work item to the next activity. Must be set if savedSetId or savedSetName is not defined.
- savedSetId - ID of saved set containing the records to be split. Must be set if id or savedSetName is not defined.
- savedSetName - Name of saved set containing the records to be split. Must be set if id or savedSetId is not defined.
- idName - Name of the property containing the comma-delimited list of internal record IDs. This should be the same as the id property value without the percent signs. The same name must be specified as the output parameter itemIds below.
- activityName - Name of the starting point activity in which the split work item(s) should be created

Outputs

- itemIds - Name of the property containing the delimited list of IDs. This should be the same as value of the input parameter idName above.
- isSplit - Work item was split if true and false if no need to split.

SQL Copy Repository Records BIC



Copies one or more records identified by a SQL query from the designated source repository to the designated target repository. By default, all attributes are copied. Optionally, some attributes may be specified to be inhibited and others may be altered by specifying the attributes by name and their new value. The process returns the number of rows read from the query and the number of records successfully copied.

Inputs

- sql_query - Defines the SQL query to execute against the Enable database. The query may contain multiple result columns. The first column must be the InternalRecordId of the source repository record. If updating records, the second column must contain the InternalRecordId of the target record, otherwise it must be null. By requiring the target record's InternalRecordId to be specified gives maximum flexibility in how the source and target records are connected. If the target InternalRecordId is not defined (i.e., null), the record will be created (if the copy should only update existing records, the query must not return any rows where the second column is null. Any additional columns can be used to alter the values being overridden as defined in attrName and attrValue. NOTE: To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '%something%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.

- sourceRepoName - The source repository name
- targetRepoName - The target repository name
- createRecord - Create a new record if 'true'. Otherwise, update an existing record as identified by the second result column returned by the query. If the second column is null, a new record will be created.
- enableTrigger - Activate trigger on target repository if true
- copyAttributes - Pipe-delimited list of attributes to be copied from the source record. If empty, include all attributes (except if any are defined in filterAttributes). If defined, filterAttributes is ignored.
- filterAttributes - Pipe-delimited list of attributes to be filtered out of the copy (i.e., not transferred from the source record. Only valid if copyAttributes not defined.
- validateAfterCopy - Validate the records after they have been copied if Yes (default is No)
- attrNameN - Name of attribute to be updated. If this attribute is transferred from the source record, it will be overridden by the corresponding value.
- attrValueN - Value for attribute to be updated. This may include any combination of literals, work item property references, or SQL result column references..

Outputs

- copyStatus - Results of copy operation (SUCCESS or FAIL)
- copyMessage - Detailed error message if copyStatus is FAIL
- sql_num_rows - Number of rows returned by the SQL query
- copy_num_rows - Number of rows copied

SQL Create Saved Set BIC



Creates a saved set based on the InternalRecordIds returned from the designated SQL query.

Inputs

- sql_query - Defines the SQL query to execute against the Enable database that returns a single column - the InternalRecordId of each record to be added to the saved set. Work item properties can be referenced by surrounding them with percent signs. NOTE: To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '\\%something\\%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.
- repositoryName - Name of the repository in which the Saved Set is to be defined. The InternalRecordId values must reside in this repository.

- savedSetName - Name of the saved set to be created. This needs to be unique or already defined in the designated repository. If the saved set name is defined in a different repository, this operation will fail with an error.
- deleteSavedSet - Delete the saved set if it already exists if Yes. If no, the records found will be added to the existing saved set.
- createTemporarySavedSet - Flags the saved set as temporary (which means it won't be visible in the UI) if YES.

Outputs

- isCreated – The saved set was successfully created if true;
- errorMsg - Details for failure if saved set was not created
- sql_num_rows - Number of rows returned by the SQL query
- savedSetId – ID of the saved set.

SQL Create Work Item BIC



Creates a work item on a workflow at the designated starting point activity with the specified work item properties for each row returned in the designated SQL query. If the query returns 0 rows, then no work items will be generated.

Inputs

- sql_query - Defines the SQL query to execute against the Enable database. Work item properties can be referenced by surrounding them with percent signs. NOTE: To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '%something%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.
- userName - Login of the user for which to create the work item. Blank means to use the default defined in Enterworks.properties
- workflowName - Name of the main workflow in which the work item resides
- activityName - Name of the activity in which the work item should be found
- workItemName - Optional name for the work item
- workItemDescription - Optional description for the work item
- batchSize - Number of records to process before delaying. A value of zero (or blank) means no batch delay will be applied.
- batchDelayMinutes - Number of minutes to delay processing after each batch has been processed

- copyProperties - Copy all work item properties from the current work item to the new work item if YES. These properties may be overwritten by the subsequent properties listed below
- wipNameX - Name of a work item property to be set (* in first property = copy all properties)
- wipValueX - Value of a work item property to be set (* in first property = copy all properties)

Outputs

- isCreated - Work items were successfully created if true");
- errorMsg - Details for failure if work item was not advanced
- sql_num_rows - Number of rows returned by the SQL query
- workItemSendCount - Number of work items sent successfully.
- workItemFailCount - Number of work items that failed to be sent.

SQL Repo Data Copy BIC



Copy data from the results of a SQL query to a target repository by generating a SQL export file using the specified query and then importing the data into the target repository using the designated import template. This BIC would typically be used when moving data from a request repository to the actual data repository.

Inputs

- sql_query - Defines the SQL query to execute against the Enable database. Work item properties can be referenced by surrounding them with percent signs. NOTE: To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '\%something\%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.
- sourceDirectoryName - Fully qualified path to directory where source file is to be placed.
- sourceFileName - Base name for source file (timestamp and extension will be added to the name)
- targetRepoName - The target repository name
- targetRepoImportTemplateName - The import template name of the target repository
- keepRepoValues - Retain repository values if value in file is empty if Yes;
- validateAfterCopy - Validate the records after they have been copied if Yes (default is No)

Outputs

- repoDataCopyStatus - Results of e-mail send operation (SUCCESS or FAIL)
- repoDataCopyMessage - Detailed error message if email.send.status is FAIL
- sql_num_rows - Number of rows returned by the SQL query

SQL Send Email BIC



Send an e-mail message to one or more recipients for each row returned in the specified SQL query. The recipient(s) can be any combination of e-mail addresses, Enable user logins, or Enable Groups. If the query returns 0 rows, then no e-mails will be generated.

Inputs

- **sql_query** - Defines the SQL query to execute against the Enable database. Work item properties can be referenced by surrounding them with percent signs. NOTE: To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '\%something\%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.
- **email_from** - Optional from address. If empty, global default will be used.
- **email_to** - Comma-delimited list of any combination of e-mail addresses, Enable user logins, or Enable Group names. Keywords surrounded by percent signs will be translated into the corresponding work item property values. Keywords surrounded by double-pipe characters will be translated into the named query result column.
- **email_subject** - Subject line for e-mail. Keywords surrounded by percent signs will be translated into the corresponding work item property values. Keywords surrounded by double-pipe characters will be translated into the named query result column.
- **email_body** - Body text for e-mail. Keywords surrounded by percent signs will be translated into the corresponding work item property values. Keywords surrounded by double-pipe characters will be translated into the named query result column.
- **email_attachment** - Fully-qualified path to file to be attached to the e-mail message. Keywords surrounded by percent signs will be translated into the corresponding work item property values. Keywords surrounded by double-pipe characters will be translated into the named query result column.
- **batchSize** - Number of records to process before delaying. A value of zero (or blank) means no batch delay will be applied.
- **batchDelayMinutes** - Number of minutes to delay processing after each batch has been processed.

Outputs

- **email_send_status** - Results of e-mail query send operation. SUCCESS if there were no errors, even if no e-mails were generated due to the query returning no rows. FAIL if there were any errors with executing the SQL or sending any e-mails.
- **email_send_message** - Detailed error message if email.send.status is FAIL

- sql_num_rows - Number of rows returned by the SQL query
- email_send_count - Number of e-mail messages sent successfully.
- email_fail_count - Number of e-mail messages that failed to be sent.

SQL Set Properties BIC



Sets properties in a work item based on the results (first row only) of a query.

Inputs

- databaseName – identifies the database to be queried: EPIM (default) or EPX
- sql_query - Defines the SQL query to execute against the Enable database. Work item properties can be referenced by surrounding them with double-pipe characters. NOTE: Percent signs are used to denote a work item property reference. To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '\%something\%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.
- inOutNameX - Name of the work item property to be updated with the corresponding in/out value.
- inOutValueX - Value to store in the corresponding in/out property. Result columns can be referenced by surrounding the column name with double-pipe characters.

Outputs

- sql_num_rows - Number of rows returned by the SQL query
- set_property_status - Indicates whether set property operation was successful (SUCCESS) or failed (FAIL).
- set_property_message - Reason for failure.

SQL Update Repository Records BIC

Updates one or more repository records identified by the ID in the first column of each result row from the designated query. If the ID is null, a new record will be created using the specified data (that may or may not be derived from the query results).

Inputs

- sqlQuery - SQL Query identifying the records to be updated. The first column must be the InternalRecordId of each record to be updated or null if a new record is to be created. additional columns may be returned and referenced in the name/value pairs by surrounding the column names with double-pipe characters (e.g., ||myResultColumn||). NOTE: To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '\%something\%'. If referencing a work item

property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.

- repositoryName - Name of repository in which the records are to be updated/created.
- enableTrigger - Activate trigger on target repository if true
- validateRecord - Validate the record after the update if true (default)
- attrNameX - Name of attribute to be updated.",
- attrValueX - Value for attribute to be updated. This can reference work item property names (surrounded by '%'), query result column names (surrounded by '| |'), literal values, or any combination.

Outputs

- ID - Delimited list of IDs of records that were created or updated ('0' means error)
- updateStatus - Results of update operation (SUCCESS or FAIL)
- updateMessage - Detailed error message if updateStatus is FAIL
- sqlNumRows - Number of rows returned by the SQL query
- updateNumRows - Number of repository records updated
- createNumRows - Number of repository records created

SQL Workflow Notification BIC



Executes a SQL query and generates a work item for each result row on the Notify workflow to generate a notification work item event.

Inputs

- sql_query - Defines the SQL query to execute against the Enable database. Each of the properties below can reference SQL column results by surrounding the column names with double-pipe characters. NOTE: Percent signs are used to denote a work item property reference. To use percent signs in the actual SQL query (e.g., wildcard in LIKE statement), precede each with a backslash character. For example: WHERE myColumn LIKE '\%something\%'. If referencing a work item property and the value may contain single quotes, surround the property reference with curly braces to ensure any single quotes are escaped. For example: WHERE myColumn LIKE '{%myProperty%}'. The curly braces must be inside the single quotes AND the property value cannot have any curly braces itself.
- notifyRecipient - Group or login name of the recipient for the notification
- repositoryName - Name of the repository containing the record represented by the work item.
- repositoryId - ID of the repository containing the record represented by the work item
- workItemId - ID of the work item for which the notification is to be generated
- taskName - Work Item Task (e.g. Notify)
- taskRole - Value of a work item property taskRole to be set
- taskDescription - Value of a work item property taskDescription to be set

- taskStatus - Value of a work item property taskStatus to be set
- workflowCommentHistory - Value of a work item property workflowCommentHistory to be set
- workItemCreationDate - Value of a work item property workItemCreationDate to be set
- batchSize - Number of records to process before delaying. A value of zero (or blank) means no batch delay will be applied.
- batchDelayMinutes - Number of minutes to delay processing after each batch has been processed.

Outputs

- sql_num_rows - Number of rows returned by the SQL query
- notifySentCount - Number of notices sent
- notifySent - All notify work items were successfully created if true
- errorMsg - Details for failure if notify work item was not created

Sync EPX Users with Enable BIC



Synchronizes EPX users with Enable based on common Enable Groups/EPX Roles. In order for users to be synched, the EPX Pool Rule role must have the exact same spelling as the Enable Group. This process will create users in EPX if they do not exist, assign EPX Roles to that group to match Enable or remove EPX roles from a user if not assigned to the corresponding Group in Enable. This callout BIC relies on theDB settings in the Enteworks.properties file.

Inputs

- None.

Outputs

- messages - Detailed messages of what operations took place.

Unlock Repository Record BIC



Unlocks a repository record (or set of records identified by a saved set) for workflow.

Inputs

- repositoryName - Name of repository containing record(s) to be unlocked.
- ID - Internal ID of the repository record to be unlocked.
- savedSetId – (TBD) Name of saved set containing the records to be unlocked
- savedSetName - (TBD) Name of saved set containing the records to be unlocked.

Outputs

- isUnlocked - Repository records have been unlocked if 'true'
- errorMsg - Details of why the record(s) have not been unlocked. Identifies each item by primary key.

UnlockWorkflowBic



This UnLock Work Flow Bic unlock a workflow. The input parameters include: unLockName, unLockAction (Unlock/Detach), workItemId

Inputs

- unlockName - Specify a unlock name.
- unlockAction - Select a unlock action from the selection: Unlock or Detach
- workItemId - ID of the work item to be unlock

Outputs

- unlockActionStatus - Results of unlockAction (SUCCESS or FAIL)
- errorMessages - Detailed error message if unlockAction is FAIL

Update Repository Record BIC



Updates a repository record identified by the ID or designated primary key (which does not have to be the repository primary key). If the createRecord flag is set, a new record will be created.

Inputs

- repositoryName - Name of repository in in which a record is to be updated/created.
- ID - Internal ID of the repository record to be updated (0 or blank if none).
- pkNameX - Attribute name for primary key (actual or alternative) of record to be updated
- pkValueX - Attribute value for primary key (actual or alternative) of record to be updated.
- createdRecord - Create a new record if 'true'
- enableTrigger - Activate trigger on target repository if true
- validateRecord - Validate the record after the update if true (default)
- attrNameX - Name of attribute to be updated
- attrValueX - Value for attribute to be updated.

Outputs

- ID - ID of record that was created or updated ('0' means error)

- errorMsg - Detailed message why the record could not be created or updated (if ID is '0')

Update User Group Assignments BIC



Adds and/or removes Enable Groups to/from Enable for the designated user. Requires synchronization with EPX be performed afterwards if user participates in workflow.

Inputs

- userLogin - Delimited list of logins to/from which group(s) are to be added/removed.
- addGroupX - Name of group to be added.
- removeGroupNameX of group to be removed..

Outputs

- groupAssignmentChanged - The expected number of Group assignment changes were made if SUCCESS, otherwise FAIL
- groupAssignmentErrors - Details of why the group assignment(s) failed.

Validate Repository Record BIC



Validates the repository record identified by the ID parameter, primary key attributes, or the records in the designated saved set. Optionally lists the attributes that must not have validation errors in order to be considered valid. Otherwise, requires the entire record to pass validation.

Inputs

- repositoryName - Name of repository containing record(s) to be validated.
- ID - Internal ID of the repository record to be validated.
- validationLevel - Validation level to be applied where A is the highest and E is the lowest. Selection of a level implies inclusion of all lower levels. Note: Enable may have the validation levels relabeled. Consult the sharedConfig.properties file to determine the appropriate level by letter.
- useShortMessages - Strip out validation level and rule name from error messages if true.
- attributeNameX - Name of attribute that must be valid if entire record does not have to pass validation. If no attributes are listed, then the record may not have any validation errors

Outputs

- isValid - Repository records are valid if 'true'

- errorMsg - Details of why the record(s) are not valid. Identifies each item by primary key and the condition(s) that failed.
- errorMsgHtml - Details of why the record(s) are not valid in HTML table format. Identifies each item by primary key and the condition(s) that failed.

Workflow Notification BIC



Generates a work item on the Notify workflow to generate a notification work item event.

Inputs

- notifyRecipient - Group or login name of the recipient for the notification.
- repositoryName - Name of the repository containing the record represented by the work item.
- repositoryId - ID of the repository containing the record represented by the work item.
- workItemId - ID of the work item for which the notification is to be generated.
- taskName - Work Item Task (e.g. Notify)
- taskRole - Value of a work item property to be set
- taskDescription - Value of a work item property to be set
- taskStatus - Value of a work item property to be set
- workflowCommentHistory - Value of a work item property to be set
- workItemCreationDate - Value of a work item property to be set

Outputs

- notifySent - Notify work item was successfully created if true
- errorMsg - Details for failure if notify work item was not created