2020

WINSHUTTLE

Best Practices for EnterWorks Deployments and Migrations

Revised 6/28/2020

WINSHUTTLE

Winshuttle Best Practices

EnterWorks Deployment and Migration

Table of Contents

WINSHUTTLE

EnterWorks Deployment and Migration – Best Practices

Introduction

The EnterWorks platform has inherent capabilities to support the migration of the solution and solution configurations from one environment to another. Winshuttle's recommended best practice is to master configuration changes in a single environment. For example, profile changes should be made only in the Dev environment and then migrated to test and then to the prod environments. Allowing changes to profiles in the prod and UAT environment requires additional steps to determine how to merge the changes.

Similarly, Winshuttle's best practice is to master data in only one environment and migrate the data from the single environment, typically test, and them migrate that data to the other environments.

Following these principles simplifies the migration process and allows a customer to rely on the delivered migration functionality for the objects listed below:

- Association Groups
- Attribute Industries
- Attribute Security Filters
- Code Sets
- Code Set Folders
- Data sources
- Export Templates
- Export Template Folders
- File Definitions
- Groups
- Hierarchies
- Hierarchy Folders
- Home Page Configurations
- Import Templates
- Profiles
- Record Security Filters
- Repositories
- Repository Folders
- Sequences
- Style Maps
- Style Map Folders
- Taxonomies
- Taxonomy Folders
- Transmission Options

- Users
- Category Attribute Associations (included when assigned Profiles are migrated)
- Link Relationships (included when Repositories are migrated)
- Object Security (included when associated objects are migrated)
- User Preferences (included when Repositories are migrated)
- CodeSet, Taxonomy, and Hierarchy folders (selected alongside the corresponding objects)
- Validation Rules (included when Profiles are migrated)

However, we understand that for a variety of reasons, customers may choose to allow mastering data in multiple environments. Much of this document describes techniques that can be used to determine out of sync scenarios between environments and provides guidance on how to identify and resolve those discrepancies.

Overview

This document describes the recommended "Best Practices" for deploying changes from one EnterWorks environment to another. It covers the following topics:

- **Deployment Definition** – detailed description of what deployment is in the context of an EnterWorks implementation.
- **Deployment Process** – general steps that are typically followed for a deployment
- **EnterWorks Migration** – detailed information regarding the use of the EnterWorks Migrate Out/In tools
- **Deployment Tools** – list of tools used to facilitate executing deployments and their details.

Deployment Definition

A deployment is defined as a set of changes to be applied to an EnterWorks environment (referred to as the "Target Environment") that result in a change of behavior in that environment. Most of the time, a deployment originates from another EnterWorks environment (referred to as the "Source Environment") but may be generated separate from EnterWorks. The majority of the EnterWorks implementations include two or three environments: DEV, QA, and PROD (in two-environment configurations, the DEV and QA are combined into one environment in such cases, the environment will be simply referred to as DEV). The general practice is to develop and test changes in the DEV environment, generate a deployment package to deploy those changes to the QA environment and re-deploy the same (or revised) deployment package to the PROD environment, with the ultimate goal being the deployment to PROD goes smoothly and results in a functioning system with the desired changes.

For implementations with only two environments, there is no opportunity to test a deployment so there is greater risk that problems arise with the deployment procedure or with the resulting changes to the PROD environment. In such cases, greater care must be taken with the creation of the deployment and with comparing the source and target environments after the deployment to ensure all changes have been deployed.

Each deployment may be comprised of one or more of the following components:

- **Copied files** – these are files that are replaced in-whole in the target environment, such as JSP pages, template configuration files, JAR files, etc.
- **File Updates** – these are changes that are made to existing files, such as Enterworks.properties and sharedConfig.properties
- **EnterWorks Migration** – this is an EnterWorks Migration Out file generated from the Source Environment that typically only contains the changes that are to be moved to the Target Environment.
- **EPX Migration** – this is an EPX Migration Out file generated from the Source Environment that typically contains the modified workflows and their subflows that fully replace the same workflows and subflows in the Target Environment.
- **EnterWorks Repository Exports** – these are data files that are used to update records in configuration repositories in the target environment. Configuration repositories include DAMConfig, DAMVariants, Scheduled Imports, Scheduled Exports, Scheduled Promotions, CN_Registry, etc. Repository data is not included in EnterWorks Migrations so the configuration repository data must be exported from the Source Environment to be imported in the Target Environment. Sometimes this configuration data includes environment-specific settings (e.g., URL for a DAMVariant, FTP server location for a Scheduled Export, etc.). These settings must be localized in the files prior to importing into the Target Environment.
- **SQL Scripts** – these are files containing SQL scripts that need to be applied to the EPIM database in the Target Environment. Examples include staging tables for syndications, data queues for processing e-mails or change notification events, stored procedures used in exports or workflows, etc.
- **Configuration Changes via the UI** – sometimes there are changes that need to be made as part of a deployment that cannot be included in a migration file but only set manually through the UI in the Target Environment. An example would be if only a subset of the changes in the source environment are being deployed to the target environment.

Deployment Process

For most deployment, the following steps are performed:

1. **Preparation** – getting ready to start a deployment.
2. **Development** – the changes are made to the DEV environment.
3. **Deployment Package Creation** – all or a subset of the changes made to the DEV environment pulled together for deployment to QA and/or PROD along with deployment instructions
4. **Deployment Execution** – applying the deployment to the target environment.

Details for each of the steps are provided in the following sections:

Preparation

When starting a new deployment (i.e., before any development work has begun), the planned contents of the deployment are known beforehand. Often, the list of changes grows during development which can happen over a long enough period of time that some of the details of what has changed are forgotten. This can be mitigated by following best practices and preparing for each deployment by

creating a Deployment Document (from a template) and creating a deployment folder on the DEV server in the EWSoftware folder.

The Deployment document should contain the following elements:

- Name and date of the Deployment
- Customer or Project the Deployment is for
- Overview that includes a summary of what the deployment contains (e.g., in bulleted list form). If the deployment includes a patch, an aggregate list of all changes since the current patch should be included with the ones potentially affecting the target implementation being highlighted
- Pre-deployment instructions, such as clearing work items from workflows being deployed
- List of files in the deployment folder on the server
- Sequence of steps to execute the deployment, grouped by physical area (some or all of the following):
    - Backup current configuration
    - Copy Files
    - Install Patches
    - Update Configuration Files
    - Update EPIM Database
    - Update EnterWorks Data Model
    - Update Configuration Repositories
    - Update EPX Workflows
- Sequence of steps to verify successful deployment
- Sequence of steps to roll back the deployment should it become necessary

In general, the more detail that is included in the deployment document the more likely the execution of the deployment will be successful.

Every time a new change is made in DEV for the deployment, the deployment document should be updated to cover that change. This practice takes some effort to develop into a habit, but it will pay dividends in the long-run – especially if the development period is spread over a larger amount of time.

If the development is a group effort, the deployment document should be placed in a shared location such that concurrent updates are possible. If the development is being done by a single person, the deployment document can be maintained locally.

The deployment folder on the server should contain any files that are part of the deployment. This does not include existing files that are only being modified (e.g., values in the Enterworks.properties or sharedConfig.properties file) and not replaced. A deployment script (e.g., deploy.bat) should be defined in the same folder to move the files to the appropriate locations and run any deployment scripts (e.g., the DeployServicesJar.bat and CopyEpimJarsToEPX.bat). Instead of manually copying the files to the target locations, any updates to the files should be placed in the deployment folder and the deploy.bat script run to copy them for you. This accomplishes several things: first, it ensures that the files are always placed in the correct location with the least amount of effort and second, it provides an opportunity to test the deployment process in a limited fashion in the DEV environment.

Except for the deploy.bat script, the files in the deployment folder should never be modified “in-place”. Instead, they should be modified and maintained in your source control/development environment and then copied to the deployment folder each time they are changed (and the deploy.bat script run to copy those files to the appropriate locations). This ensures that a file changes are captured in the source version control.

Since some files (such as JARs) require running of other scripts and restarting services while others (such as JSPs) can be copied while EnterWorks services are running, it may be beneficial to define several deployment scripts and have the deploy.bat call all of them. For example, the script deployFilesOnly.bat can be defined to copy the files that do not require a services restart. The script deploySqlOnly.bat can be defined to load any SQL scripts into the database. The main deploy.bat script would call these two as well as perform the operations that require a services restart. It may seem redundant to copy all files any time one of them changes, but the time required is negligible and it ensures that nothing is missed (i.e., copied to the deployment folder but then not actually deployed).

Development

At the start of a deployment, it is a good practice to conduct an initial compare between the DEV environment and PROD to make sure the development effort is starting with a good baseline. It is often the case that the customer has made data model changes directly to the PROD environment. These typically include changes to code sets, profiles, taxonomies and hierarchies. If the DEV environment is not first “refreshed” from PROD, there are two possible consequences:

1. The development and testing will be invalidated when deployed to PROD, meaning the lack of those changes could result in different behavior when testing in DEV. This of course depends on the nature of the changes and the planned development. For example, code sets not directly associated with the development pose a lower risk than changes to a Profile that is being modified for the development.
2. The deployment itself may risk a regression in PROD and undo those changes. For example, if a datatype of and attribute is changed in PROD and the same repository is updated on DEV as part of the development, the migration of the profile could result in the datatype being reverted back to the old datatype.

If the scope of the discrepancies between DEV and PROD are small, it may be quickest to simply make the same changes on DEV via the UI or import separate export files generated from PROD. The EnterWorks Migration tool should be used if there are many changes and in different areas. Similarly, changes to SQL objects (temp tables, stored procedures, views, etc.) that are likely to be maintained by the customer should be checked and if altered, they should be extracted and the source version control for the project be updated so subsequent changes are based off the newer versions. The updated scripts should then be applied to the DEV environment to get it in sync with PROD.

If there is a QA environment, it too should be updated with the changes being made to DEV to match PROD. This ensures that when the deployment to QA is executed, the outcome will match what will happen when the deployment is applied to PROD.

For the EnterWorks data model objects, configuration repositories, database objects, and EPX workflows, the compare extract SQL scripts should be used to identify the discrepancies between DEV and PROD. Most of the time it can be assumed that the values in PROD should be used to update DEV. However, sometimes the discrepancy reflects a change made to DEV that was not deployed to PROD. When there is doubt, it is best to first check with any Professional Services personnel who may have been involved with the project previously and then take the questionable discrepancies to the customer as a last resort.

Once development for a deployment has begun, it is easy to focus on just making the changes and not keeping the deployment folder and document up-to-date. It is strongly recommended to maintain both the deployment folder and the document as development progresses. This reduces the burden when it is time to create the deployment package and increases the likelihood that all aspects of the deployment have been captured and are under source control.

Deployment Package Creation

Once the development for a deployment and its unit testing in DEV, the final deployment package needs to be created. The contents of the deployment folder should be reviewed to ensure the latest versions of all files have been copied to it (which should be the case if the best practice of always updating the files their and running the deploy.bat script has been followed). The deployment document also needs to be finalized. The level of detail produced mostly depends on who will be executing the deployment.

A final compare should be done between the DEV and PROD environments. In general, the only differences between the environments should be the ones included in the deployment. However, it's possible that subsequent changes have been made directly to PROD by the customer or that DEV has parallel development that will be included in a later deployment. If differences that are not attributed to a future deployment are found, an assessment must be made as to whether those differences will interfere or overlap with the deployment. If they will, those changes should be made to the DEV environment followed by whatever regression testing is appropriate before completing the deployment package.

There are some deployment artifacts that will not be created until the deployment package is being created. This includes exports from configuration repositories, and the EnterWorks and EPX migration files. The comparison between DEV and PROD can be very helpful with ensuring the all the necessary objects are exported or migrated out of EnterWorks and EPX.

When exporting records from configuration repositories, the user preference "Deploy Export" should be used to ensure only the fields that should be migrated from one environment to another are included. If this user preference does not exist, one should be created. Most of the attributes should be included in the user preference. Ones that should not are typically auto-sequenced attributes (the IDs will likely not match between environments and are excluded from the comparison extracts) and state or timestamp type of attributes.

It's possible that the configuration repository records included in the deployment will contain environment-specific settings. For example, a Scheduled Export may have an FTP server as a target and the FTP server or target directory are different for each environment. Each of the exported repository

files should be reviewed to ensure any environment-specific values are updated to reflect the target environment.

Deployment Execution

Once the full deployment package has been created, it is ready to be executed in the QA and/or PROD environment. The deployment folder should be copied to the EWSoftware folder in the target environment. In most new environments, the file structure will be identical between environments. If there are differences, it may be necessary to add some conditional logic to the deploy.bat script to detect which environment in which it is running and copy the files to the appropriate directory. Similarly, any SQL updates will need to reference the target database (and credentials). A good practice is to copy the deploy scripts from the previous deployment to the new one as it is likely the same types of files will need to be deployed. If there is a need for conditional logic, copying the scripts also ensures refinements are made once and used repeatedly which will greatly increase the rate of successful deployment executions.

EnterWorks Migration

Most of the configuration settings within EnterWorks can be migrated from one environment to another. For non-migratable objects, updates to a target environment from a source environment must be done through the UI. Examples of non-migratable objects include:

- Server Properties
- Log level settings
- Publication Manager objects
- Configuration repositories (e.g., DAMConfig, DAMVariants, Scheduled Exports, Scheduled Imports, Promotions, CN_Registry, etc.)

The migratable objects that can be selected directly by the migration tool include:

- Association Groups
- Attribute Industries
- Attribute Security Filters
- Code Sets
- Code Set Folders
- Data sources
- Export Templates
- Export Template Folders
- File Definitions
- Groups
- Hierarchies
- Hierarchy Folders
- Home Page Configurations
- Import Templates
- Profiles
- Record Security Filters

- Repositories
- Repository Folders
- Sequences
- Style Maps
- Style Map Folders
- Taxonomies
- Taxonomy Folders
- Transmission Options
- Users

There are some migratable objects that cannot be selected directly, but get migrated when other objects are selected. These include:

- Category Attribute Associations (included when assigned Profiles are migrated)
- Link Relationships (included when Repositories are migrated)
- Object Security (included when associated objects are migrated)
- User Preferences (included when Repositories are migrated)
- CodeSet, Taxonomy, and Hierarchy folders (selected alongside the corresponding objects)
- Validation Rules (included when Profiles are migrated)

Details are provided below for each migratable object:

Association Groups

Association Groups must be migrated prior to or in conjunction with the Profiles that reference them.

Attribute Industries

Not used.

Attribute Security Filters

Attribute Security Filters must be migrated prior to or in conjunction with any repository security with them assigned.

Code Sets

Code Sets must be migrated prior to or in conjunction with any profile where they are assigned to attributes. When a code set is migrated, the entire code set is migrated to the target, including deleting codes that no longer exist in the source migration. Unlike when code set codes are changed through the UI where effected records are updated to contain the new value, the migration of a changed code set will be treated as a delete/add. Any references to the old code value must be manually updated post-migration.

Data Sources

Data sources are rarely migrated as the majority of the imports do not access a JDBC data source.

Export Templates

Export Templates must be migrated after or in conjunction with the repositories, link relationships, and attributed they reference.

File Definitions

While it is possible that file definitions exist, it's not often that File Definitions need to be migrated. Certainly, any definition having a name of "FileDef_" should not need to be migrated as they reflect temporary file definitions that did not get cleaned up automatically. The only time a File Definition needs to be migrated is if there is one associated to a Repository View Mapping but this functionality has been largely supplanted using Import Templates.

Groups

Migrating In Groups with the Overwrite option set will replace the user assignments in the target with the ones defined in the source environment (providing the target environment has the same user). It is generally not recommended to overwrite the Groups since the user assignments in each environment is likely to be very different.

Hierarchies

Hierarchies need to be migrated before or in conjunction to any other object that references them.

Home Page Configurations

Some of the home page configurations include references to actual object IDs. These IDs are likely not migratable, meaning without being updated, they will be incorrect and unexpected behavior may result. Depending upon the type of widget and its settings, it may be necessary to manually migrate changes via the UI.

Import Templates

Import Templates need to be migrated after or in conjunction with the repositories and attributes they reference.

Profiles

Attributes removed from a profile in the source environment will not be deleted from the target. Instead the Migration log file will warn that the attribute exists in the target but not the migration. This provides the opportunity to export that data and import it (after possible transformation) before deleting the attributes from the profile. But in the end, the attributes have to be manually deleted.

Profiles have the option to migrate individual attributes vs. the entire profile. For each attribute, it can be set to be added if not present or overwritten if it exists.

Profiles containing reference to a category attribute association object (for Taxonomy attributes) will also migrate the Category Attribute Association object.

Record Security Filters

Record Security Filters need to be migrated after or in conjunction with the Profiles they reference and before or in conjunction with any repository and group to which they are assigned.

Repositories

Migration of repositories includes the Attribute Properties settings (e.g., snapshot, index, default value, and JSP pop-up, link-relationship)

Repositories must be migrated in with the Overwrite option in order for any link relationship associated with the repository to be migrated. The option to select individual link relationships for migration is non-functional at the time of this writing.

Repository Folders

Repository folders must be explicitly migrated prior to or in conjunction with the repositories they contain.

Sequences

Sequences should generally not be overwritten (only added if not present), unless the data in the associated repository is also being migrated. Otherwise the sequence number could end up being reset to a lower number and sequences on new records may collide with existing records.

Style Maps

Style Maps need to be migrated before or in conjunction with the Publication or Syndication Templates that reference them.

Taxonomies

Taxonomies need to be migrated before or in conjunction with the Profiles that reference them.

Transmission Options

Transmission Options need to be migrated before or in conjunction with any repository that references them.

Users

Users are generally not migrated from one environment to another. But if they are, they need to be migrated before or in conjunction with any Groups that are migrated.

Security

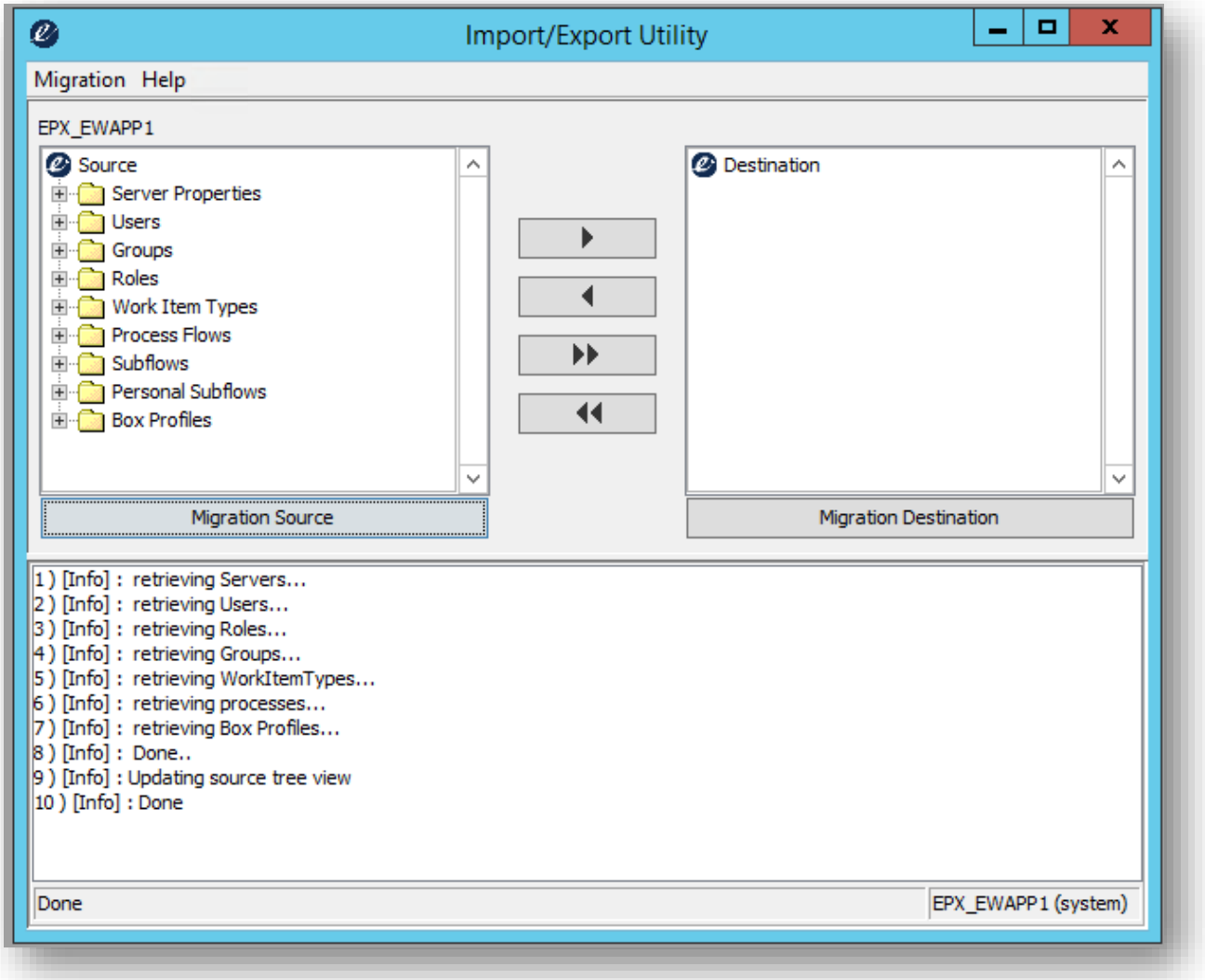
The attempt is made to migrate the security settings with each securable object. These settings are subject to the differences in user/group assignments between the source and target environments. It

may be necessary to include manual steps in the deployment procedure to ensure the security is properly set in the target. This need should be minimized if security is managed at the group level only.

When Migration link relationships, the associated repositories must be migrated (at least one of the two being referenced by the link relationship, and the Migrate In must be set to Overwrite for that repository.

EPX Migration

Migrations in EPX are managed from the EPX Design Console using its Import/Export utility:

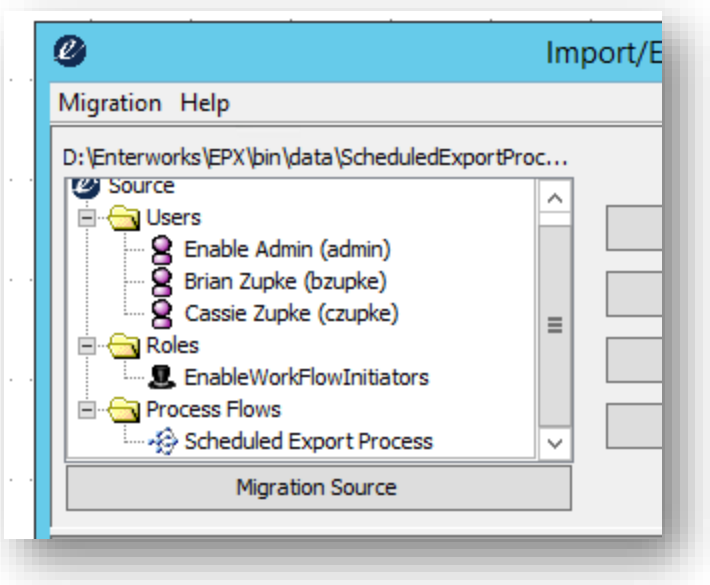


The Import/Export Utility allows for the following types of objects to be migrated:

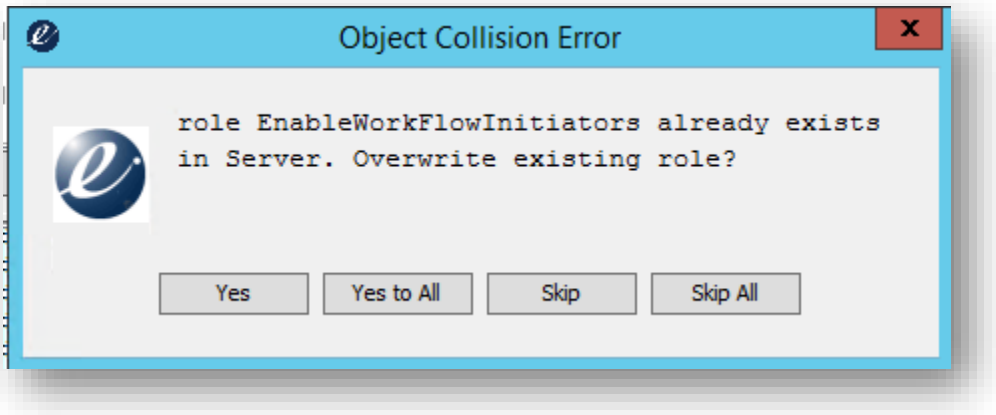
- **Server Properties** – mostly environment-specific settings that should only be migrated as a form of backup, not to transfer to the target environment

- **Users** – interactive workflows require EnterWorks users to be defined in EPX. However, since there is a process for syncing the EnterWorks users with EPX, it is generally not necessary to migrate users directly
- **Groups** – groups are predefined and are not associated with EnterWorks Groups and don't need to be migrated.
- **Roles** – roles determine who can participate in workflows at each step. Some may match EnterWorks Groups. The roles do need to be migrated but they will be included with any workflow that is migrated (if the role is not already defined in the target). Therefore, are generally not explicitly selected. If an existing role is migrated, the assigned users from the source will replace the assigned users in the target. These are often entirely different groups.
- **Work Item Types** – work item types are used to define the input fields on the Save and Send forms. They must be explicitly migrated if they don't exist on the target. If an existing work item type is updated on the source, it may need to be explicitly migrated, separately with the Overwrite option when the collision is reported.
- **Process Flows** – the process flows need to be deleted on the target before migrating replacements. There is an intended function to support migrating existing workflows with active work items (and transferring those work items from the old flow to the new version), but it is currently non-functional. Any active work items need to either be completed or purged before the old workflow can be deleted.
- **Subflows** – similar to Process Flows, the subflows must also be deleted before replacements can be migrated. But in order to delete a subflow, all Process Flows and subflows that call it must also be deleted. There is an intended utility (created by Professional Services) that facilitates deploying new versions of Subflows with active work items but it is currently non-functional.
- **Personal Subflows** – personal subflows have the same requirements as Subflows.
- **Box Profiles** – box profiles can be used to define the columns a user sees in their active work item list on the Worklist Task Manager page (not the My Active Work Items widget)

When generating a migration file (export), the EPX Import/Export tool includes all dependency objects in the file. For example, if the Scheduled Export Process workflow is exported, the export file will contain not only the Scheduled Export Process workflow, but the EnableWorkflowInitiators role and the users assigned to that role:



When this workflow migration file is imported into a target EPX server, any dependency object that is not defined will be added automatically. If the object already exists, the EPX Migration tool will report a collision and the administrator will need to decide whether the object is to be overwritten or skipped:



Under most circumstances, the Skip All option should be selected if the listed object is a dependent object (i.e., not the workflow, subflow, or work item type, etc.) that was selected for import. This is because overwriting the existing object will completely overwrite the current settings. For objects like Roles, this means all of the users who are assigned to the role in the target environment will be replaced with the users assigned to the role in the source environment which will not likely be the same set of users. If the colliding object is a process flow or subflow, the overwrite option will actually create a new process flow or subflow with the text “Migrated” added to the name. The details for updating process flows and subflows are defined below.

One of the rare times Yes should be selected on the collision prompt is if migrating an updated work item type or migrating a Role in which the assigned users does need to be migrated to the target EPX.

When Migrating an EPX process flow and/or subflow to a target where either the workflow/subflow are new or where there are no active work items in those workflow/subflows on the target, the recommended best practices is to delete the existing process flow/subflows on the target (purging any work items first) and importing the replacements into the target.

If the processflow and/or subflow being migrated already exists in the target system AND has active work items that cannot be purged, the existing workflows need to be renamed (to avoid a collision with the new ones being migrated) and the new ones imported. The work items in the old versions will continue processing until completion but new work items will only be created on the new workflow. If the work items in the old workflow need the behavior defined in the new versions, those work items must be terminated and recreated in the new workflow(s).

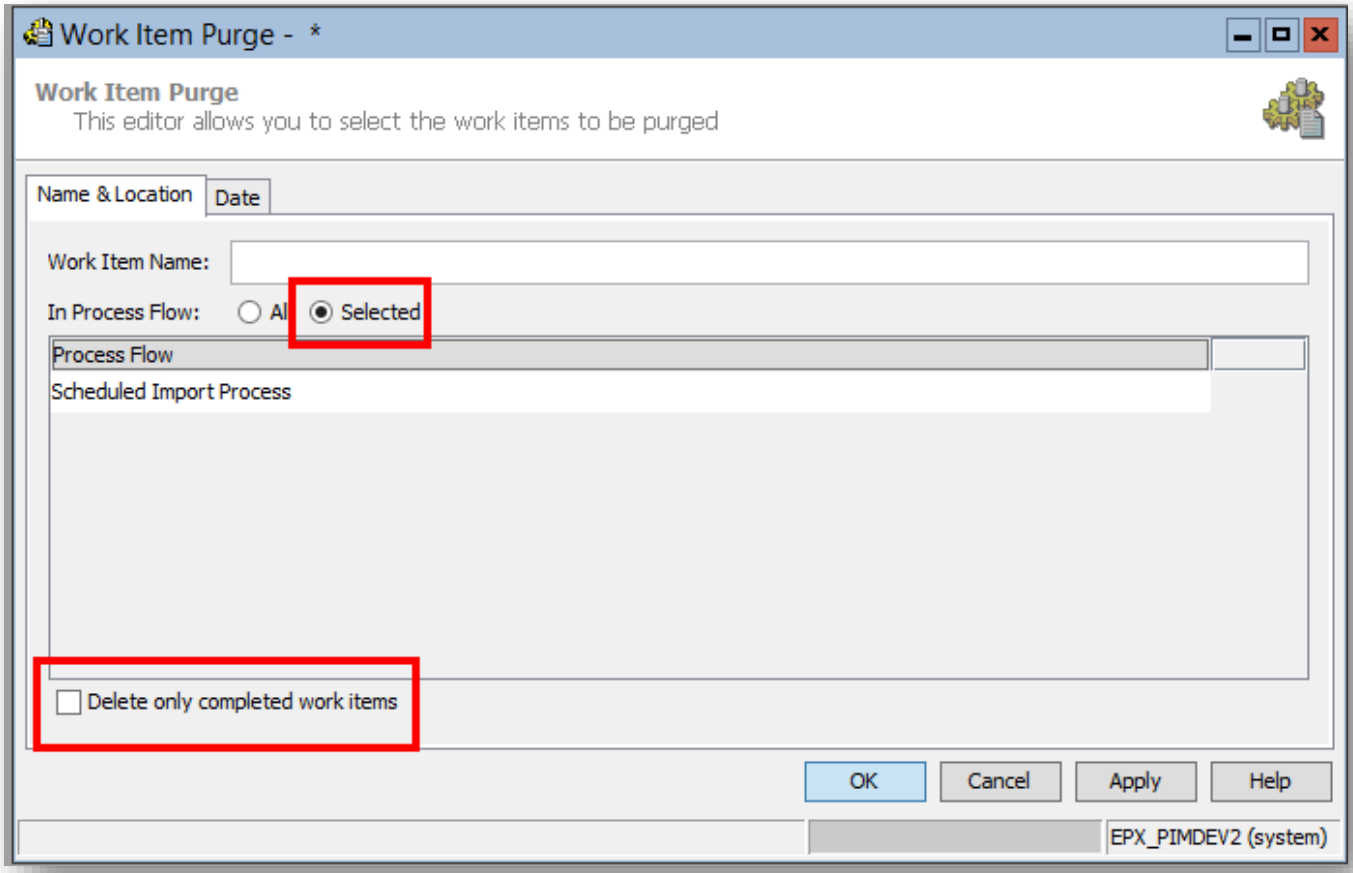
Migrating EPX Process Flows and Subflows – No Active Work Items

If the target EPX server does not contain the Process Flows and Subflows being migrated, or there are no active work items in the Process Flows in the target EPX, perform the following steps to migrate the Process Flows and Subflows from the source:

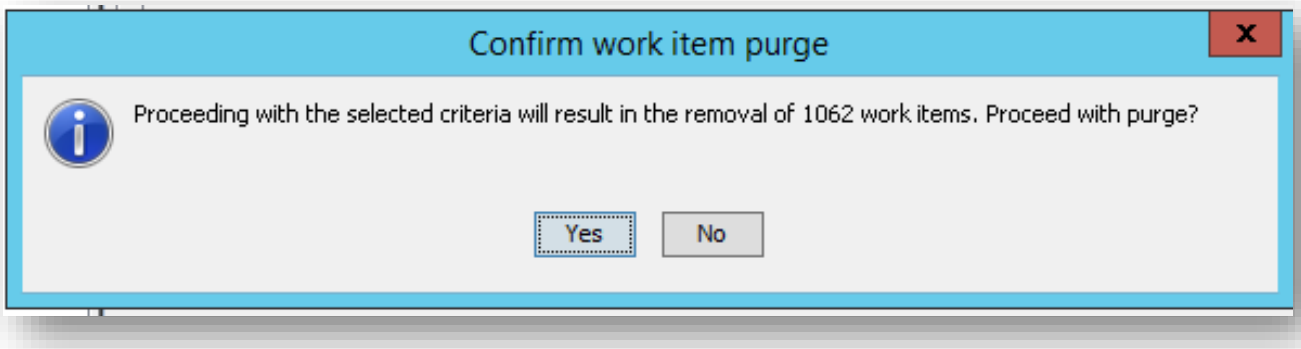
1. Open the EPX Design Console and log in as user system.
2. Expand the **Process Modeling->Process Flows** folder:
3. Compile a list of process flows to be migrated. If a subflow needs to be migrated, include all process flows that call the subflow:

Workflow	Description
<list of workflows to be migrated>	

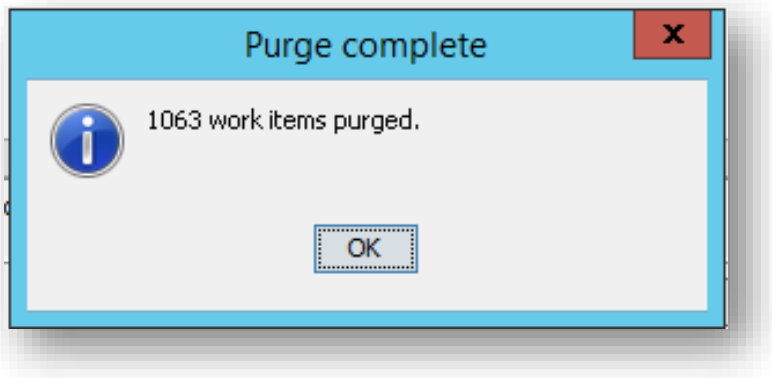
4. Perform the following steps for each workflow in the list above>
    a. If the workflow is not defined, skip to the next one in the list above
    b. Right-click on the workflow and select **Purge Work Items…** from the pop-up menu.
    c. Check the **Selected** radio button for In Process Flow
    d. Uncheck the **Delete only completed work items** checkbox:



e. Click **OK**. A confirmation prompt appears:

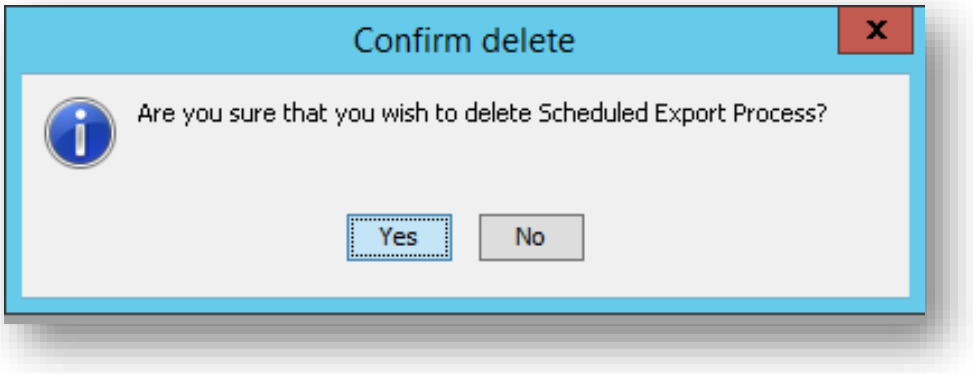


f. Click **Yes**. The work items will be purged and a completion prompt appears:



g. Click **OK**.

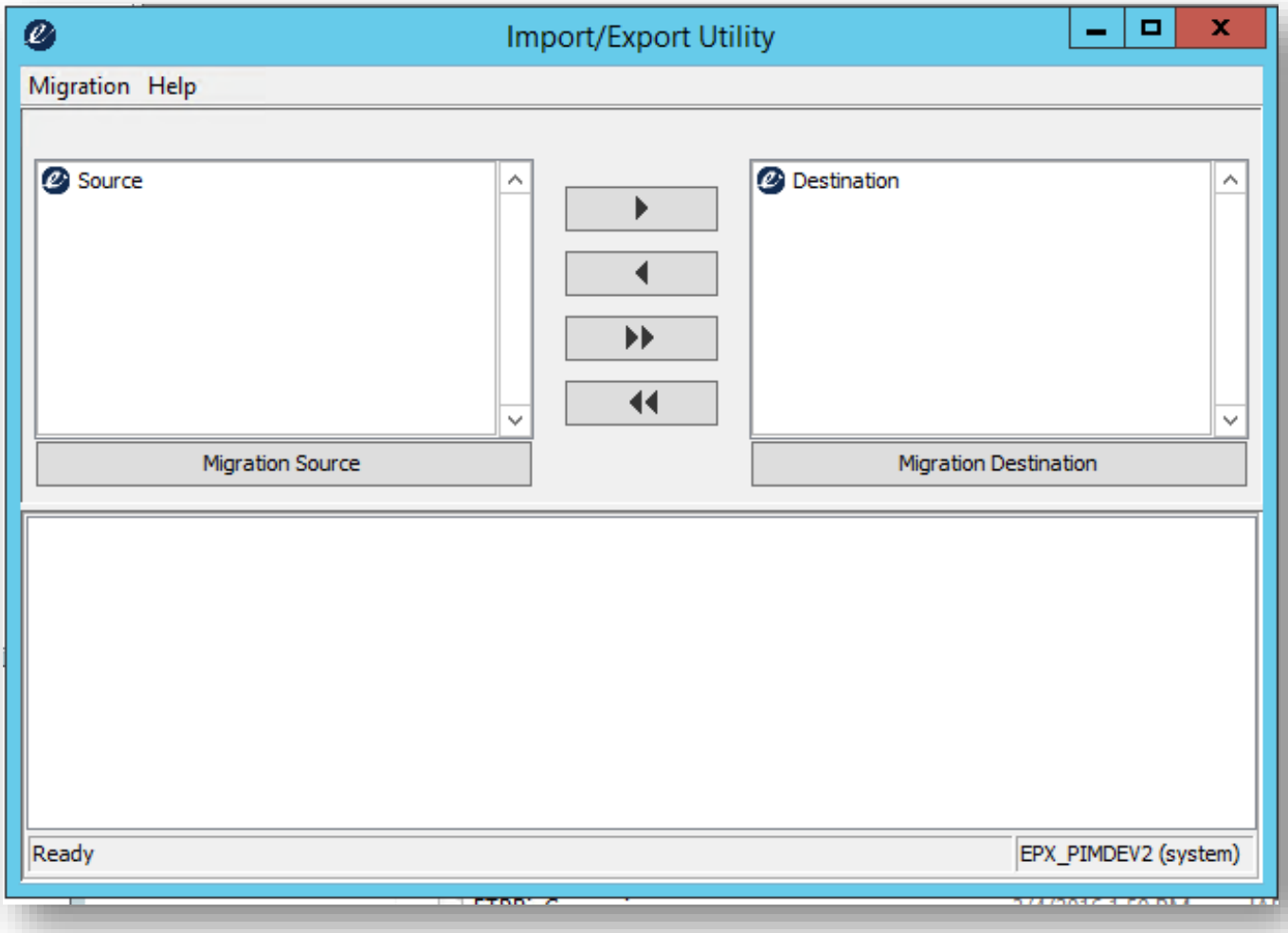
h. Immediately right-click on the workflow again and select **Delete** from the pop-up menu. A confirmation prompt appears:



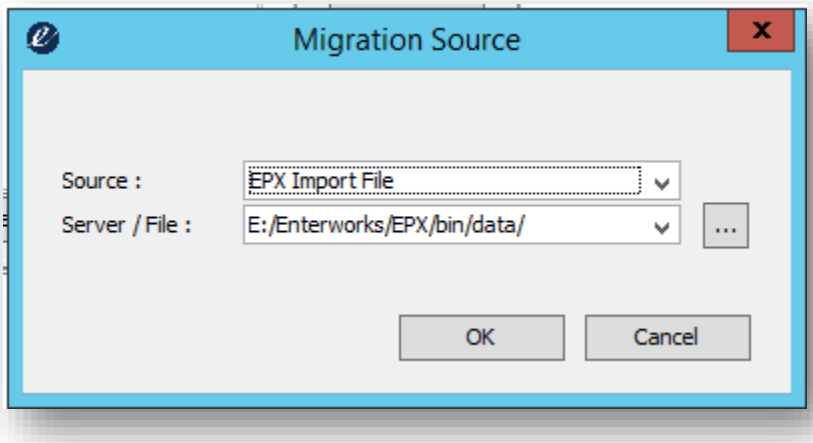
i. Click **Yes**. Note: If a dependency prompt appears, it means a work item was created on the workflow between the purge and the delete attempt. If this happens, repeat the purge/delete steps above.

j. Repeat the above steps for the next workflow in the list.

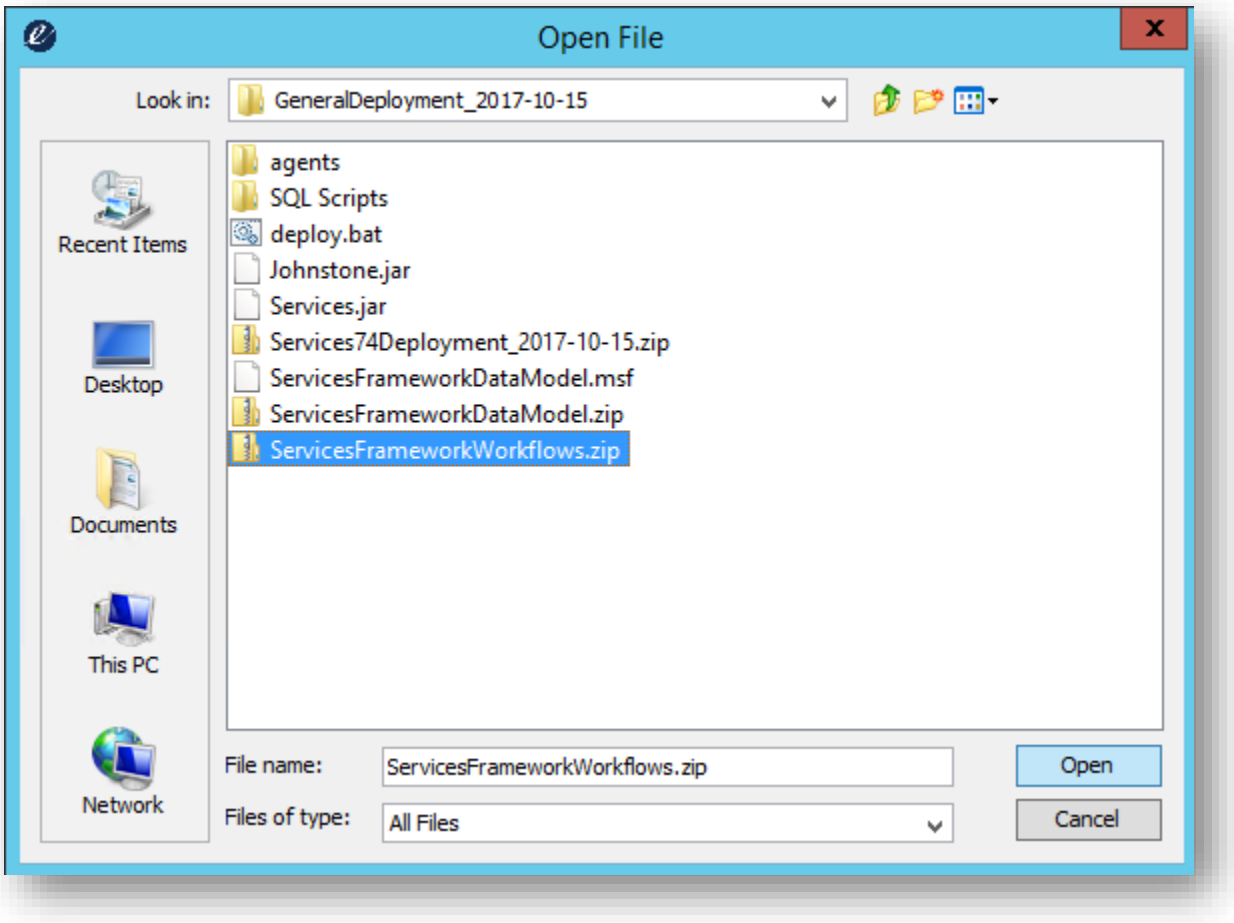
5. Select the menu option **Tools->Import/Export** menu option (if the option is disabled, click on the EPX_<server> entry in the navigation tree first). The Import/Export Utility window appears:



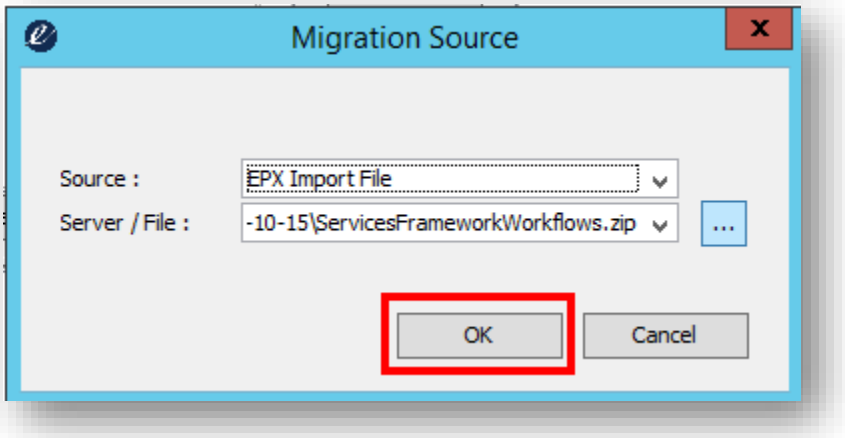
6. Click the **Migration Source** button. The Migration Source prompt appears.
7. Select **EPX Import File** for the Source and click the **…** button:



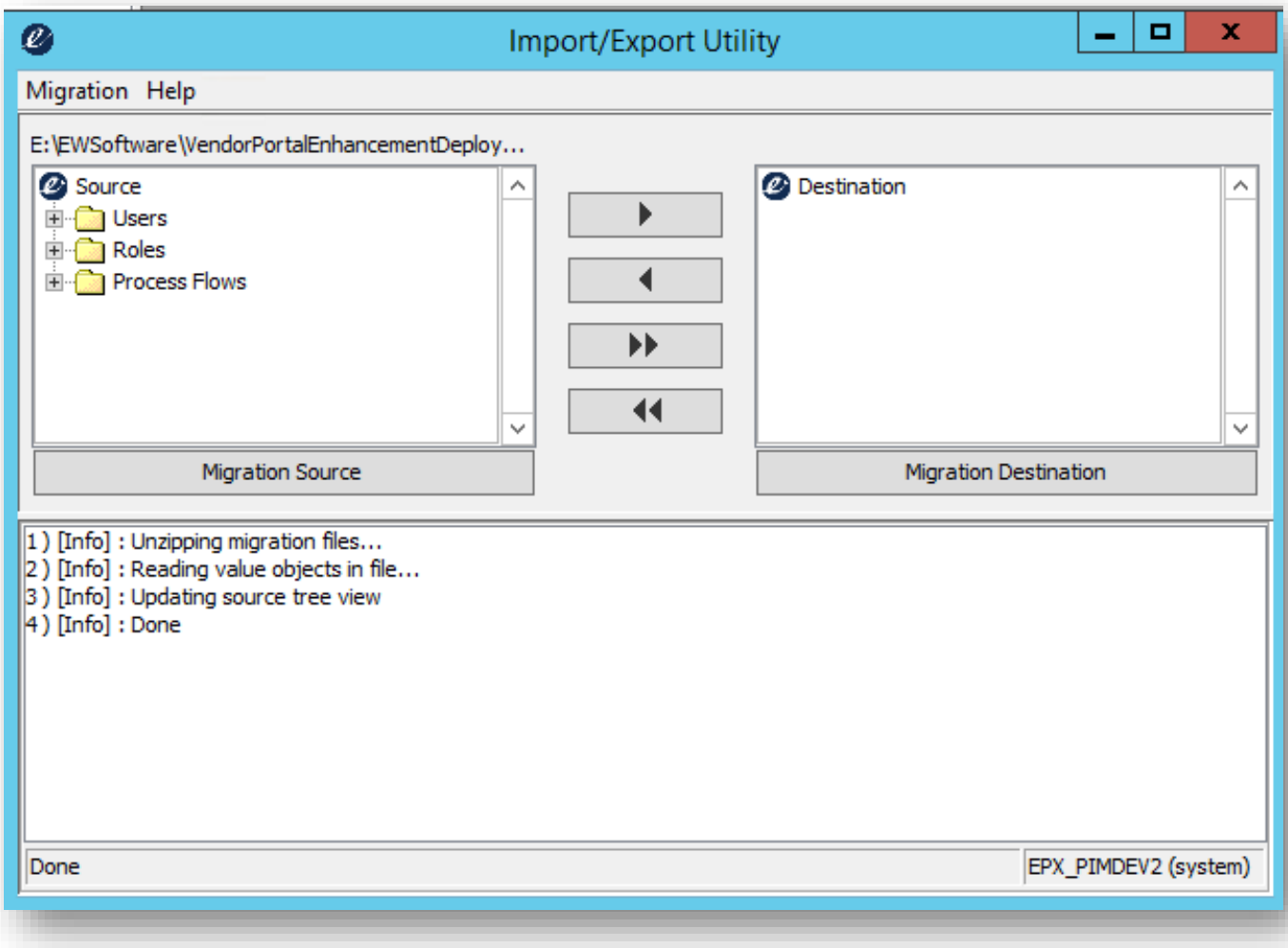
8. A file browser appears.
9. Select the EPX workflow zip file generated from the source EPX and click **Open**:



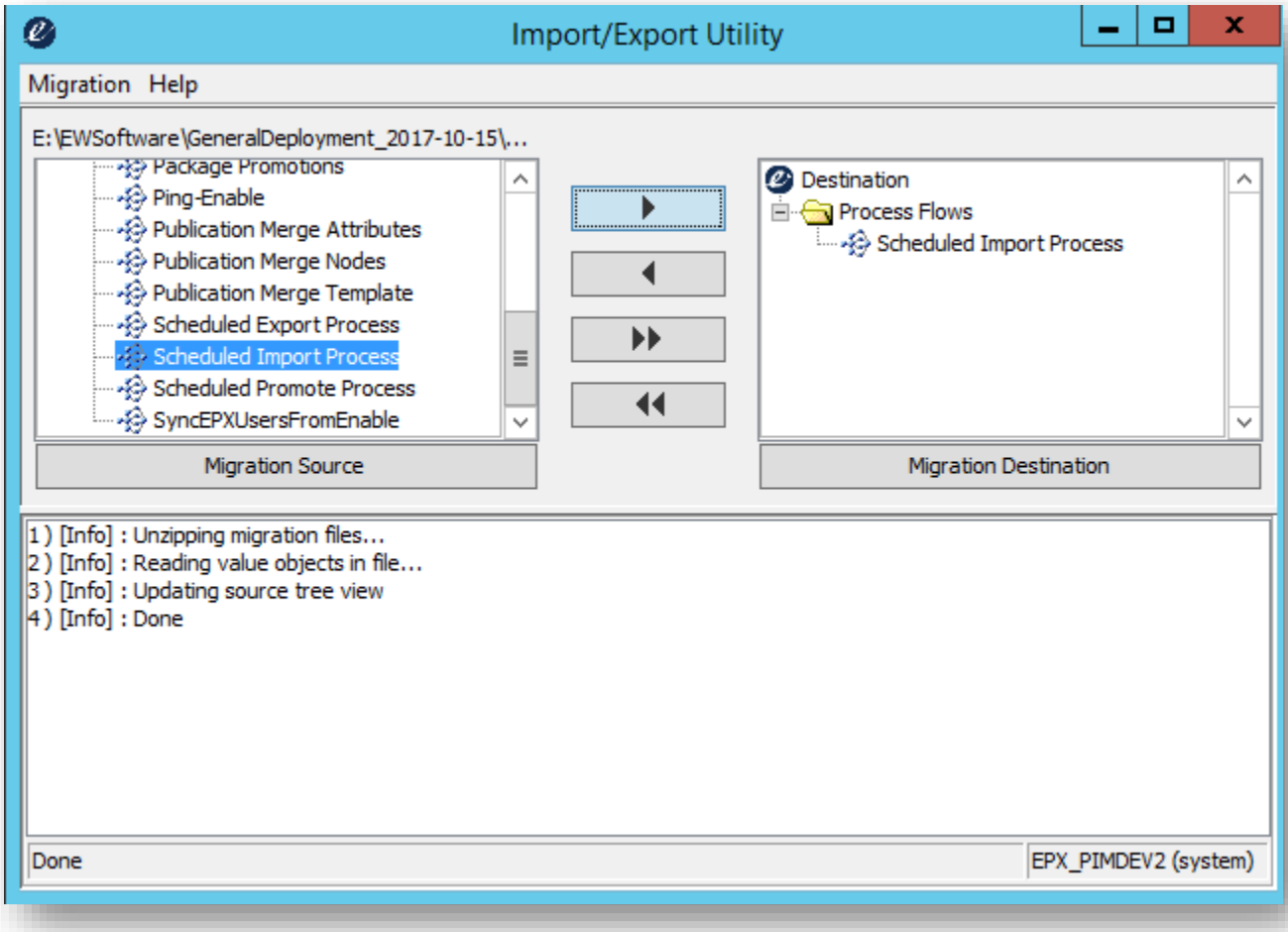
10. Click **OK**:



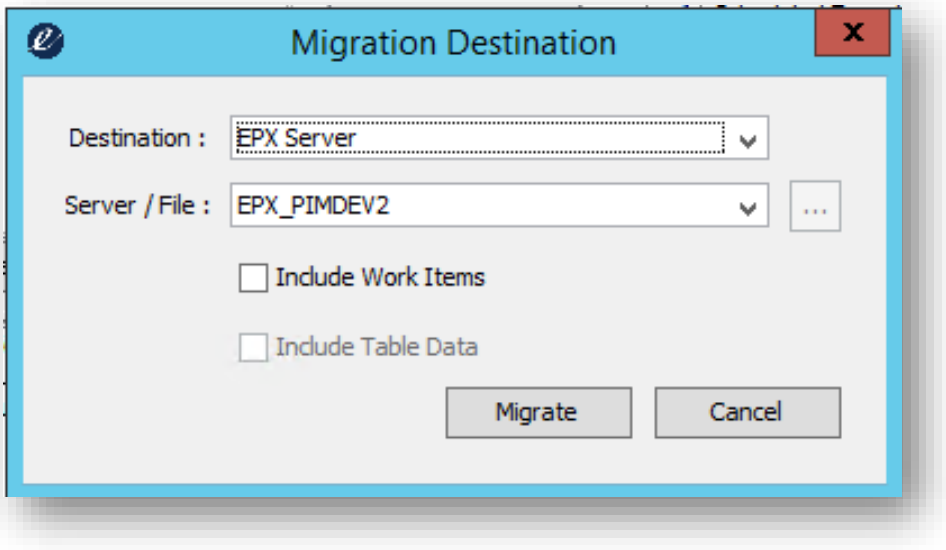
11. The EPX Migration file is loaded into the Source window:



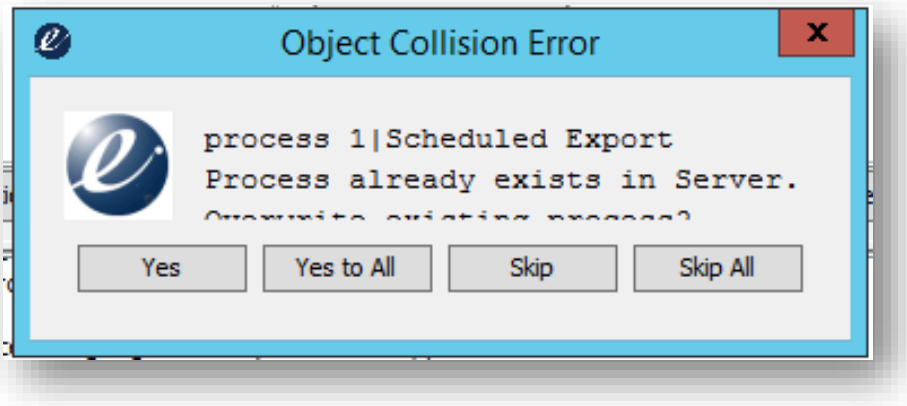
12. Expand the **Process Flows** and **Subflows** folders
13. Click on the each process flow and/or subflow and click the single right-arrow button. The Process Folder appears in the Destination list:



14. Click the **Migration Destination** button. The Migration Destination prompt appears:



15. Click **Migrate**. The Two workflows are re-created. Eventually a Collision Error prompt appears (the message will be different):



16. Click **Skip All**. The migration processing completes.
17. If any of the workflows/subflows invoke the Email BIC activity and the target EPX has different mail server requirements (e.g., DEV does not use SLS or TLS security but PROD does):
    a. Edit the workflows containing the Email BIC activity
    b. Modify each Email activity.
    c. Change the security on the Outbound settings to SSL or TLS
18. Stop and restart all Enterworks Services

Migrating EPX Process Flows and Subflows – Active Work Items

If the target EPX server already contains the Process Flows and Subflows being migrated, and there are active work items in the Process Flows in the target EPX that need to be retained and completed, then the old versions of the workflows cannot be deleted from the target EPX prior to the migration of the new versions. While EPX has a work item transfer utility that is intended to transfer work items from the old versions of the workflow to the new, it is not functional. Instead, the recommended strategy is to allow the existing work items to complete their processing in the old version of the workflow (while preventing any new work items from being created) and have all new work items be created in the new version of the workflow.

If the nature of the changes to the workflow are such that the existing work items need to have the changes apply to them, then it will be necessary to manually update the workflow(s) in the target EPX instead of migrating the workflows. The Compare Extract Script for EPX can be used to ensure that all of the changes in the source have been made correctly in the target.

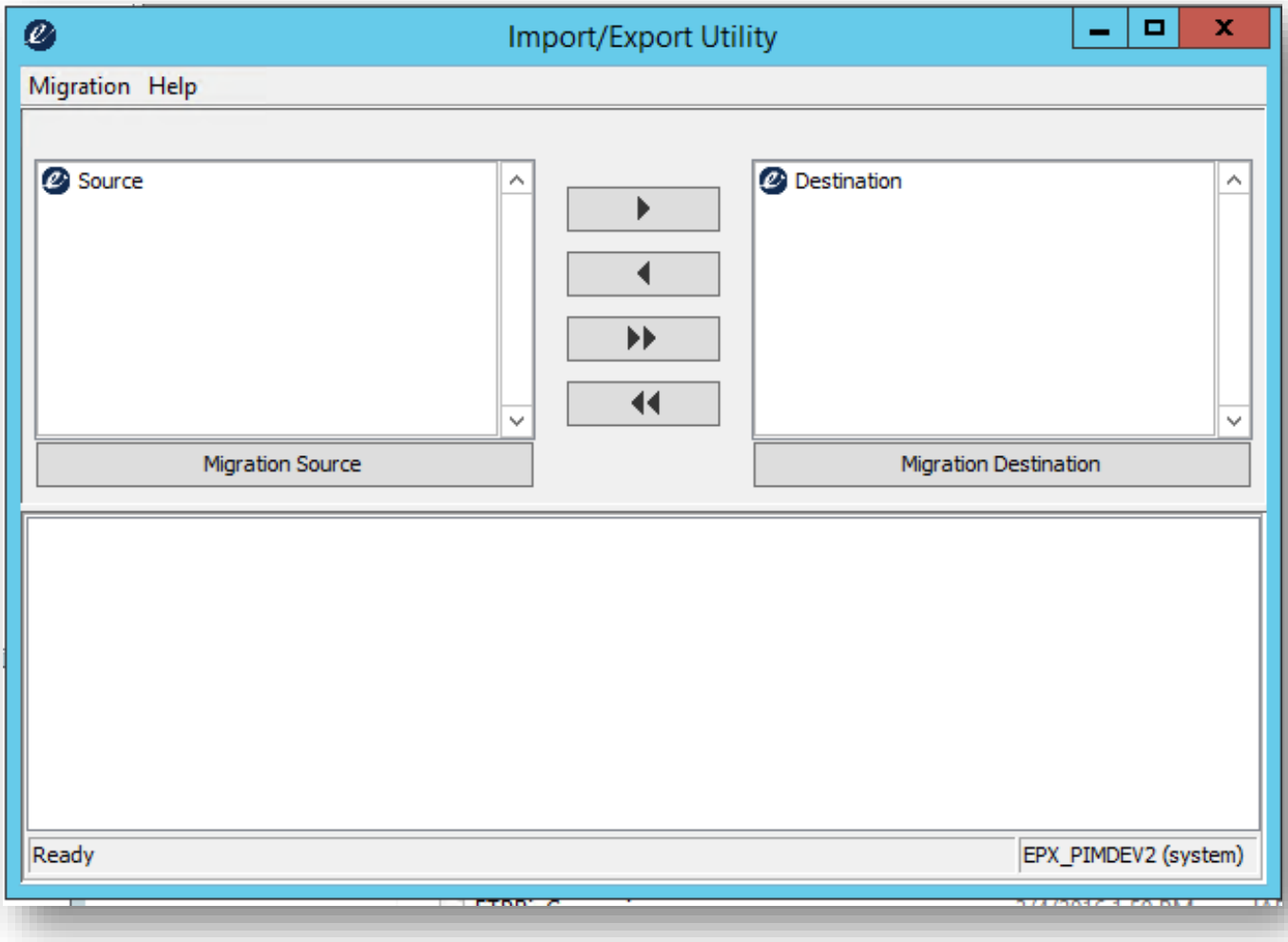
If the current work items can complete their processing in the old version of the workflow, then migrate the new EPX workflows by performing the following steps:

1. Open the EPX Design Console and log in as user system.
2. Compile a list of process flows to be migrated. If a subflow needs to be migrated, include all process flows that call the subflow:

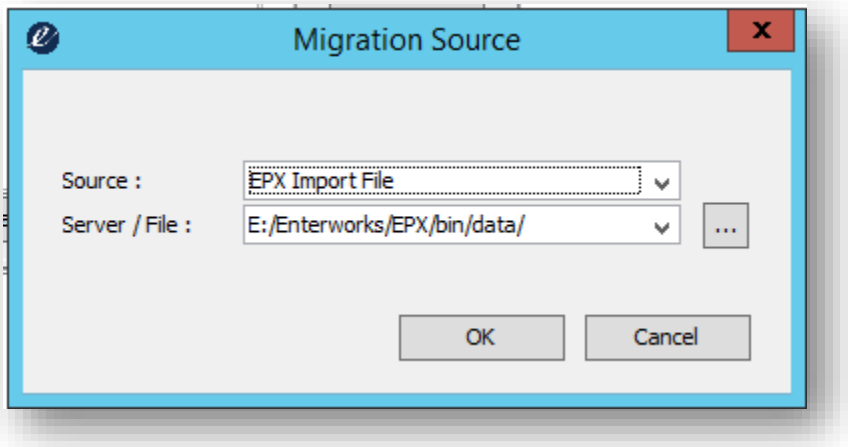
Workflow	Description
<list of workflows to be migrated>	

1. Expand the **Process Modeling->Process Flows** and **Process Modeling->Subflows** folders:
2. For each process flow and subflow in the above list, perform the following steps:
    a. Right-click on the Process Flow or Subflow and select Open from the pop-up menu.
    b. Add " OLD" to the end of the Name for the Process Flow or Subflow.
    c. Click on the Process Flow Modeler tab.
    d. For each starting point manual activity in the workflow:
        i. Double-click on the activity (or right-click on the activity and select Properties from the pop-up menu).
        ii. Click on the Details tab.
        iii. Select on the Select… button.
        iv. Select a user from the list that is not active in EPX (create a user if necessary). This effectively disables the old workflow, preventing any users from creating work items for it.
        v. Click OK.
        vi. Click OK.
        vii. Repeat the above steps for any additional starting point manual activities
    e. Click Save.
    f. Repeat the above steps for each process flow/subflow in the list above.

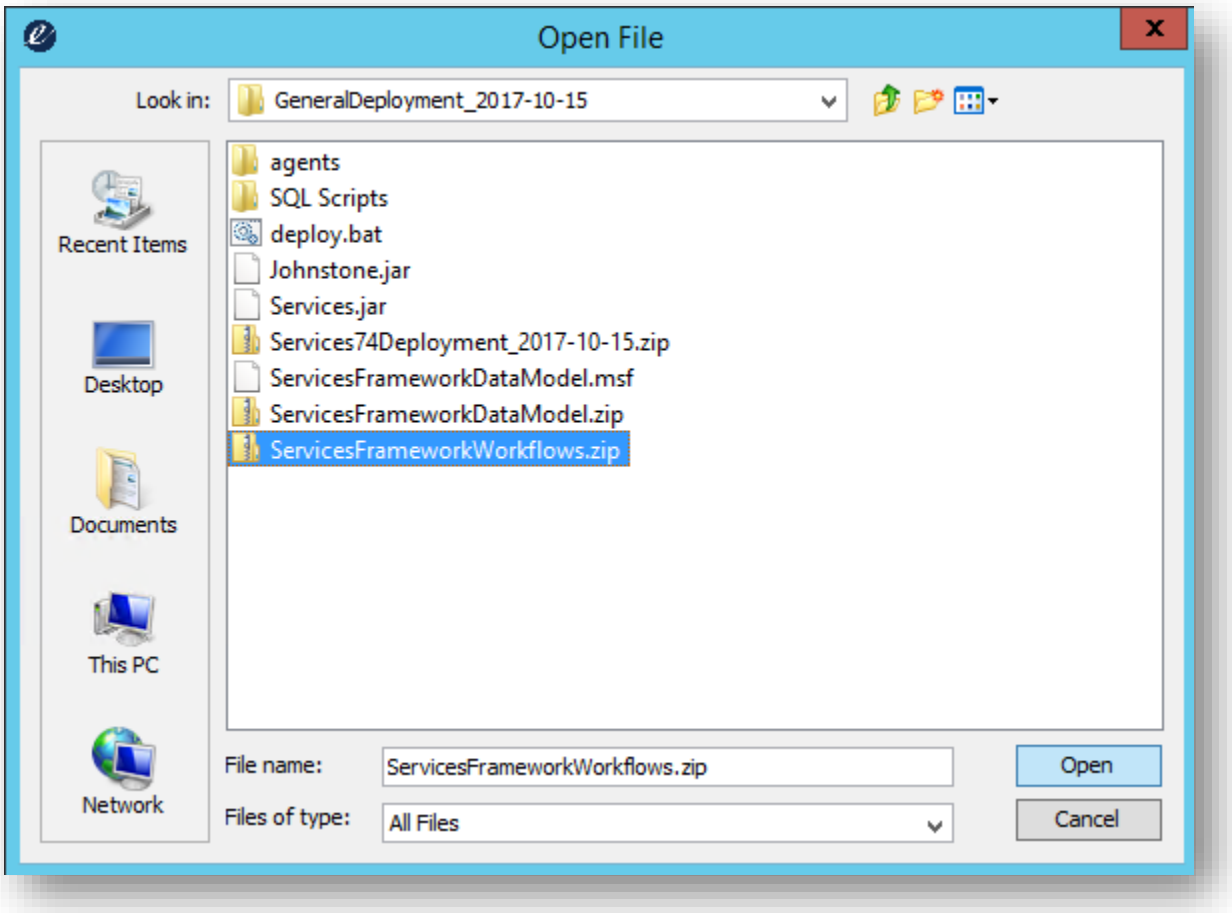
3. Select the menu option **Tools->Import/Export** menu option (if the option is disabled, click on the EPX_<server> entry in the navigation tree first). The Import/Export Utility window appears:



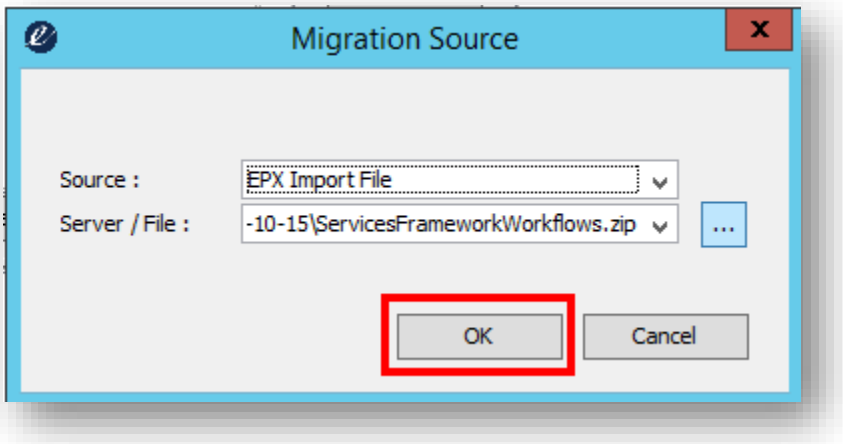
4. Click the **Migration Source** button. The Migration Source prompt appears.
5. Select **EPX Import File** for the Source and click the **…** button:



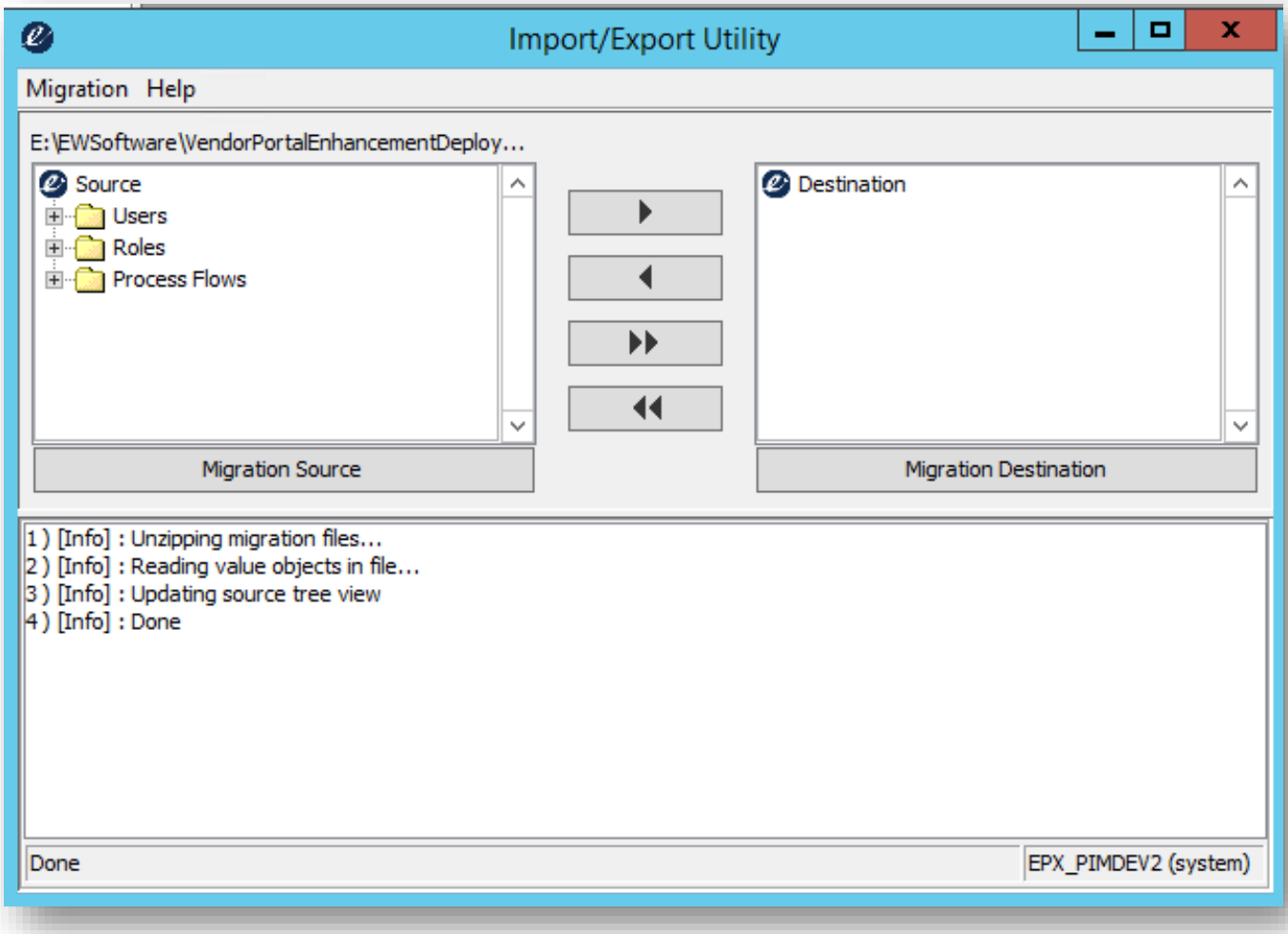
6. A file browser appears.
7. Select the EPX workflow zip file generated from the source EPX and click **Open**:



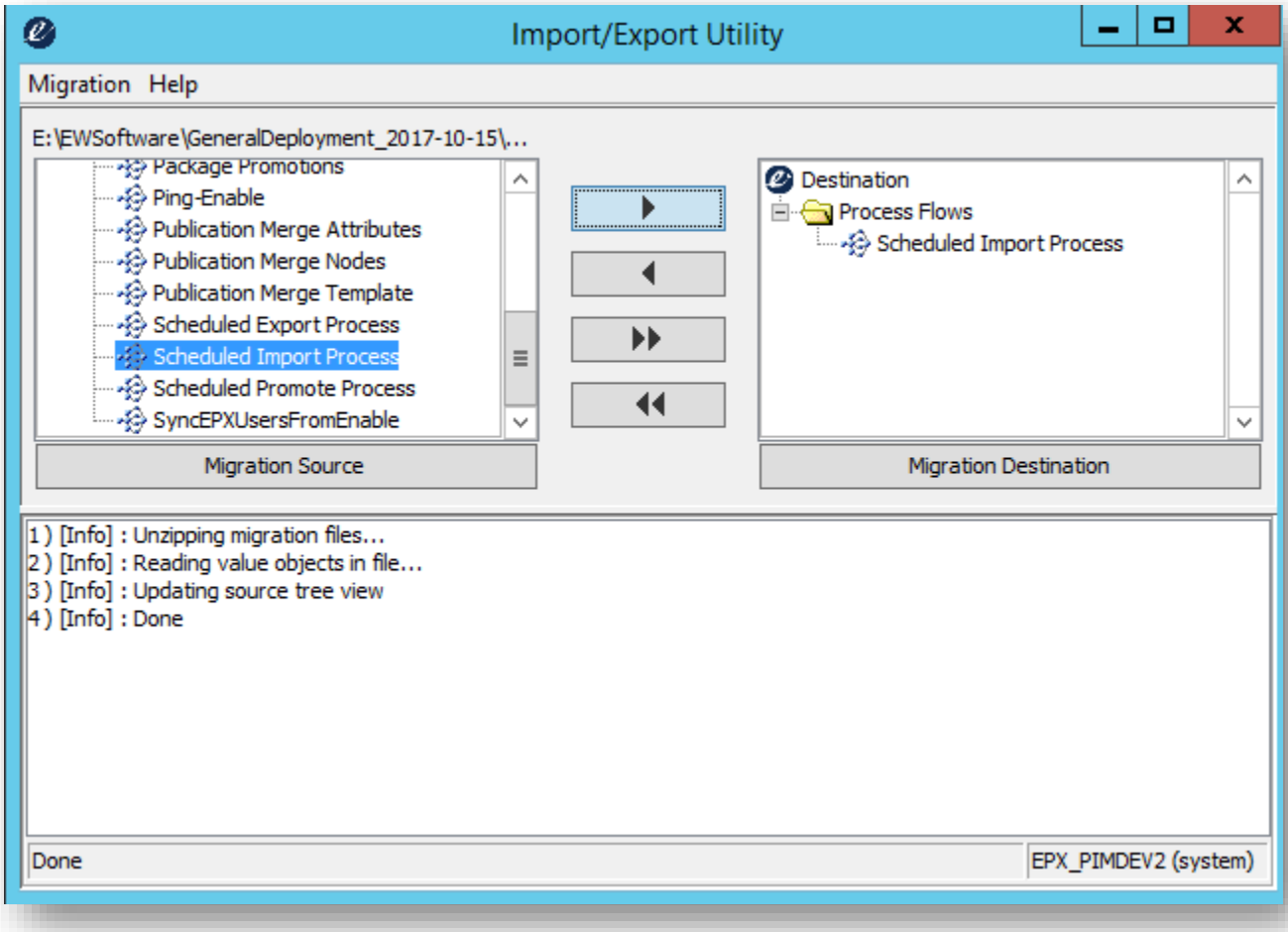
8. Click **OK**:



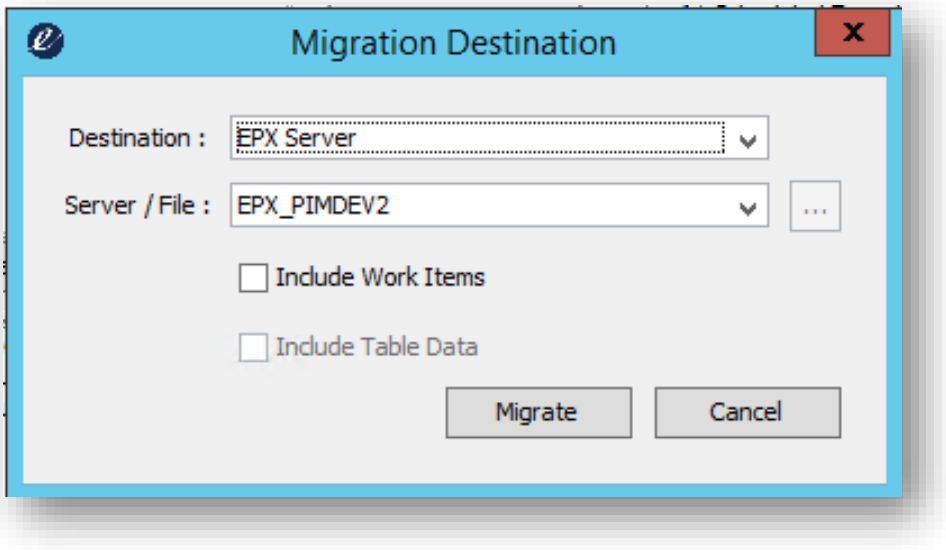
9. The EPX Migration file is loaded into the Source window:



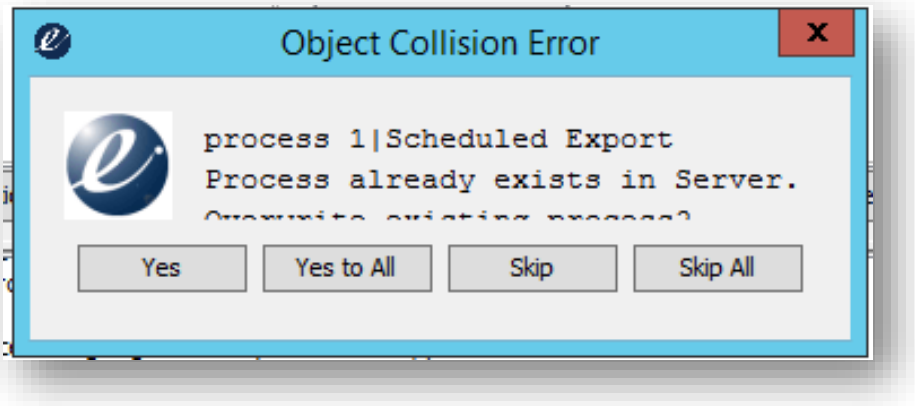
10. Expand the **Process Flows** and **Subflows** folders
11. Click on the each process flow and/or subflow and click the single right-arrow button. The Process Folder appears in the Destination list:



12. Click the **Migration Destination** button. The Migration Destination prompt appears:



13. Click **Migrate**. The Two workflows are re-created. Eventually a Collision Error prompt appears (the message will be different):



14. Click **Skip All**. The migration processing completes.
15. If any of the workflows/subflows invoke the Email BIC activity and the target EPX has different mail server requirements (e.g., DEV does not use SLS or TLS security but PROD does):
    a. Edit the workflows containing the Email BIC activity
    b. Modify each Email activity.
    c. Change the security on the Outbound settings to SSL or TLS
16. Stop and restart all Enterworks Services

EPX Migration Pitfalls, Limitations, and Workarounds

Work Item Types

Work Item Types, which are used to define submission forms must have their internal IDs match in both systems. If the Work Item Types are migrated from one EPX to another, there is no guarantee that their IDs will be the same. Similarly, if work item types are manually created in different orders within the different EPX environments, they will end up with different internal IDs. It would be best if this can be avoided by manually creating work item types in all environments at the same time and in the same order. If the work item types share the same internal IDs, then the workflows that reference them can be migrated without needing to be adjusted manually. If the internal IDs are not the same in each environment, then the incorrect submission form may end up being assigned to manual activities in the target EPX and will require manual correction. A SQL script to auto-correct the Work Item Types is to be supplied.

Work item types are not automatically migrated when referenced in manual activities. They must be explicitly selected in the Import/Export migration tool.

Repository IDs

The repository IDs that are assigned to the manual activities are migrated along with the repository names. If the same repositories have different internal IDs in the different EnterWorks environments, the references in the manual activities may be incorrect in the target system. If the different EnterWorks environments are refreshed through database backup/restore, the IDs will be identical, and migration will not be a problem. Otherwise, it will be necessary to either manually check and correct the repository IDs in each manual activity to the appropriate values or run the SQL script that corrects them automatically (recommended).

Migrating Subflows

When migrating subflows, special care is needed since they are linked to process flows and possibly other sublfows. If a subflow is migrated by itself, it is recommended that the old subflow be renamed (with Old) so the new subflow does not collide. Once the new subflow has been migrated, each calling process flow and subflow will need to be edited to point to the new version of the subflow. If the old version of the subflow has any active work items, they must be advanced to exit the subflow before the transition is made. If this is not possible or practical, the steps for migrating process flows and subflows with active work items should be followed and the process flows migrated as well.

Manual Migration

If changes to a workflow or subflow are migrated manually (i.e., the changes are made in the target EPX through Design Console editing), the Compare Extract procedure should be followed to ensure that all changes have been made and exactly match the source EPX. T

Deployment Tools

This section provides details on the tools available to facilitate deployments.

Compare Extract Scripts

The compare extract scripts will extract will generate formatted reports from SQL Server Management Studio of the various EnterWorks and EPX objects in a specific structured order to facilitate comparing two environments by comparing the corresponding extract files.

Each extract script has a recommend text field size to ensure the possible values are complete while not making the physical size (in terms of character width) excessively large. The following table lists the available scripts, their purpose, the recommended text size. The instructions for setting the text size are included below.

SQL Script	Purpose	Query Text Size
CompareEnableServerExtract_CodeSets.sql	Extracts the Code Sets, Taxonomy and Hierarchy definitions	512
CompareEnableServerExtract_Core.sql	Extracts the most common objects, including groups, profiles, validation rules, repositories, and configuration repositories.	512
CompareEnableServerExtract_ImportExport.sql	Extracts the import template, export template, syndication template, and publication template definitions	512
CompareEnableServerExtract_SearchPref.sql	Extracts the User Preferences and defined advanced searches.	512
CompareEnableServerExtract_Security.sql	Extracts the details for the attribute and repository security filters and the security settings for all groups and objects	512
CompareEnableServerExtract_ServicesFramework.sql	Extracts only the model objects associated with the Services Framework	512
CompareEnableServerExtract_SQL.sql	Extracts the tables, views and stored procedures from the EPIM database	8192
CompareEnableServerExtract_SQL_ServicesFramework.sql	Extracts the tables, views and stored procedures associated with the Services Framework from the EPIM database	8192
CompareEPXServerExtract.sql	Extracts the users, roles, and workflows from EPX	8192
CompareEPXServerExtract_ServicesFramework.sql	Extracts the workflows associated with the Services Framework	8192

When each script is executed and the results output to a file, it is recommended to follow a naming convention for those files such that the project and environment and specific content are identified using the following naming convention:

<type>_<environment>_<project>.rpt

Where:

- <type> - identifies the type of file: CodeSet, Core, ImportExport, SearchPref, Security, ServicesFramework, SQL, ServicesFrameworkSQL, EPX, ServicesFrameworkEPX
- <environment> - identifies the environment: DEV, QA, PROD
- <project> - identifies the project: CPO, ACPRO, Orgill, etc.

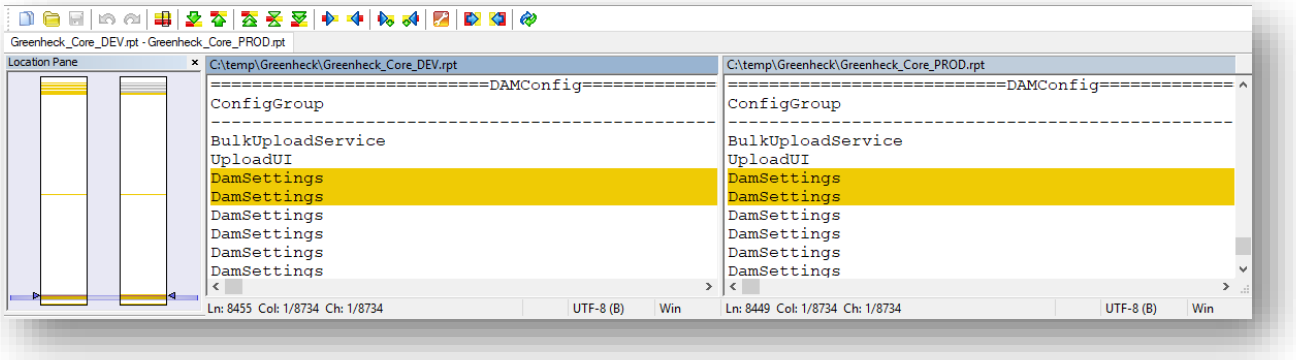
Once the files have been generated from each environment, they can be compared using a tool such as WinMerge or searched using a text editor.

Each file will generate sections that begin with a series of equal signs and identify the section, followed by more equal signs. For example, the Core report includes the following sections:

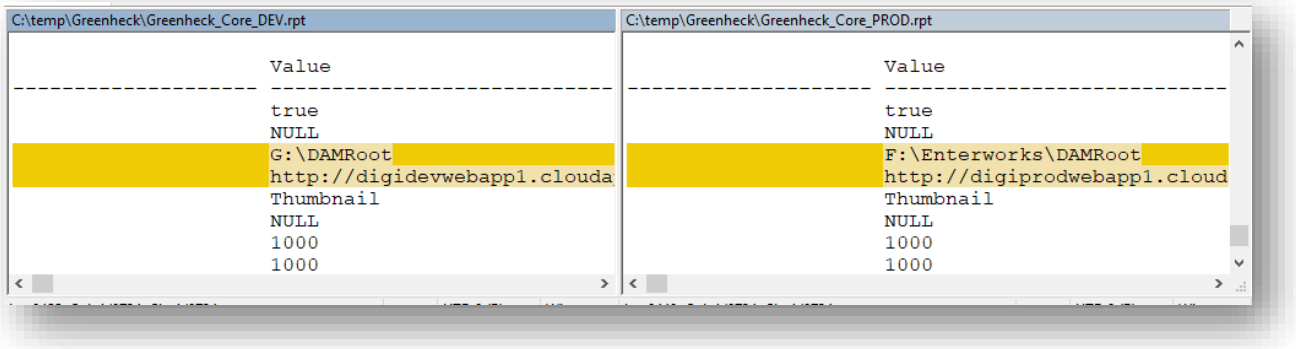
```
==========Groups==========
==========Group Capabilities==========
==========Group Home Page Widgets==========
==========Profiles==========
==========Validation Rules==========
==========Link Relationships==========
==========Repository Folders==========
==========Repositories==========
==========Repository Attributes==========
==========Code Sets==========
==========Scheduled Exports==========
==========Scheduled Imports==========
==========Package Promotions==========
==========Promotions==========
==========Shortcuts==========
==========CN_Registry==========
==========MQ_Registry==========
==========DAMConfig==========
==========DAMVariants==========
==========Automated Sort==========
==========Publication Merge==========
==========Sequences==========
==========Server Properties==========
==========Transmission Options==========
```

Each section has a set of columns, many self-identifying (since the report can be quite large) with fixed positions (determined by the query text size of 512 or 8192). The rows in each section are ordered so differences are more easily identified by WinMerge.

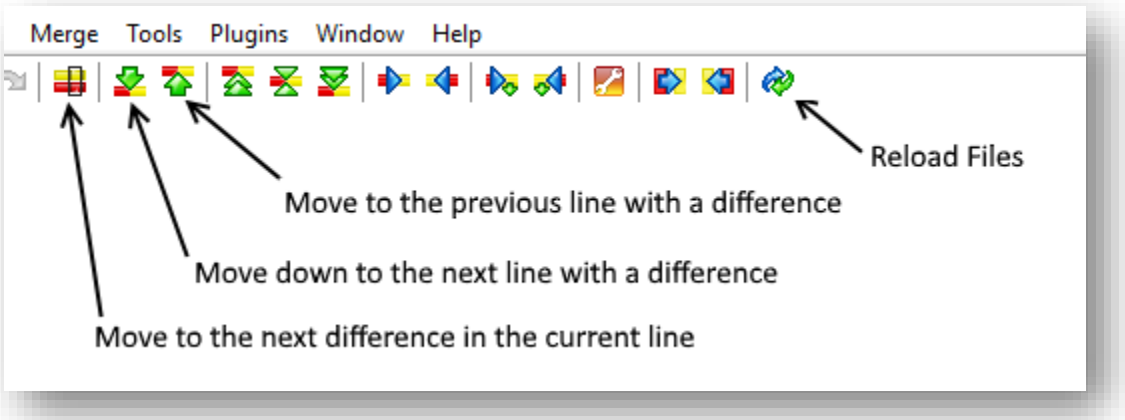
When two files for the same report (but different environments) are loaded into WinMerge, the differences between then are highlighted:



The Location Pane graphically shows where there are differences in the file. Clicking on a location will move the file view panes on the right to that location. Lines that have differences will be shaded. When the horizontal scrollbar is moved, the contents in both panes move together. The actual differences in the line will be shaded a different color:

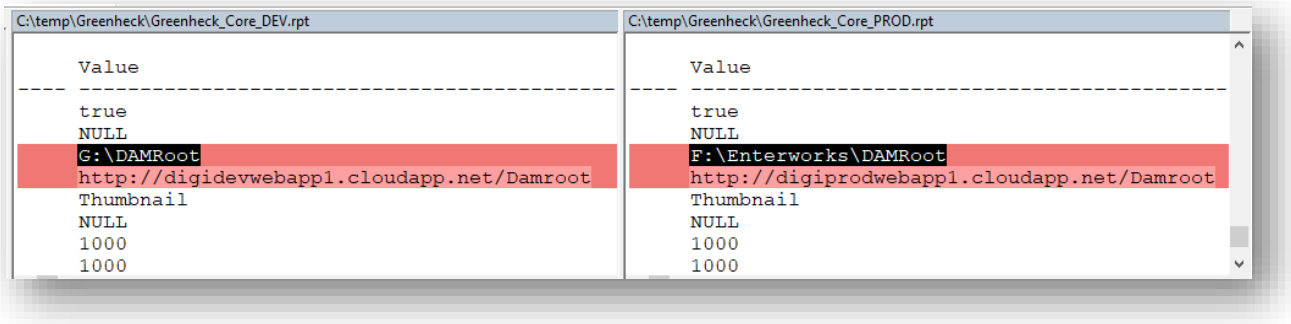


The toolbar has several buttons that help navigate through the changes:



For details on the other icons, consult the WinMerge documentation or online help.

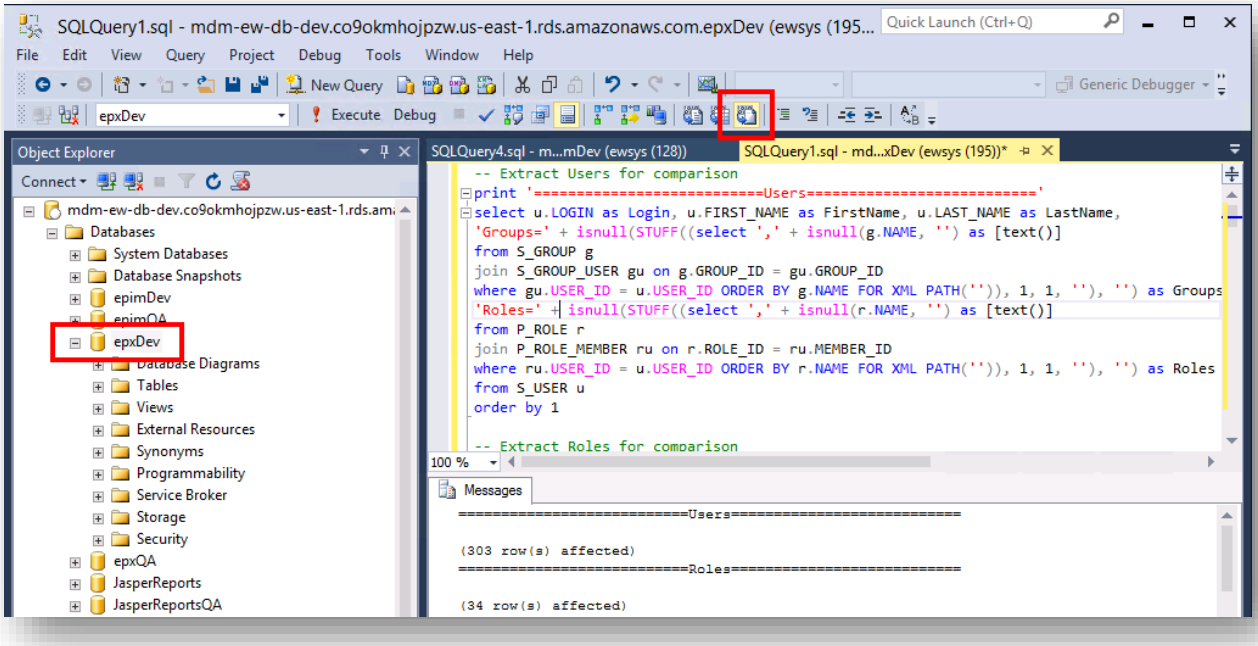
The following screen shot shows how the differences appear when using the navigation buttons to move up or down and to find the next difference in the current row:



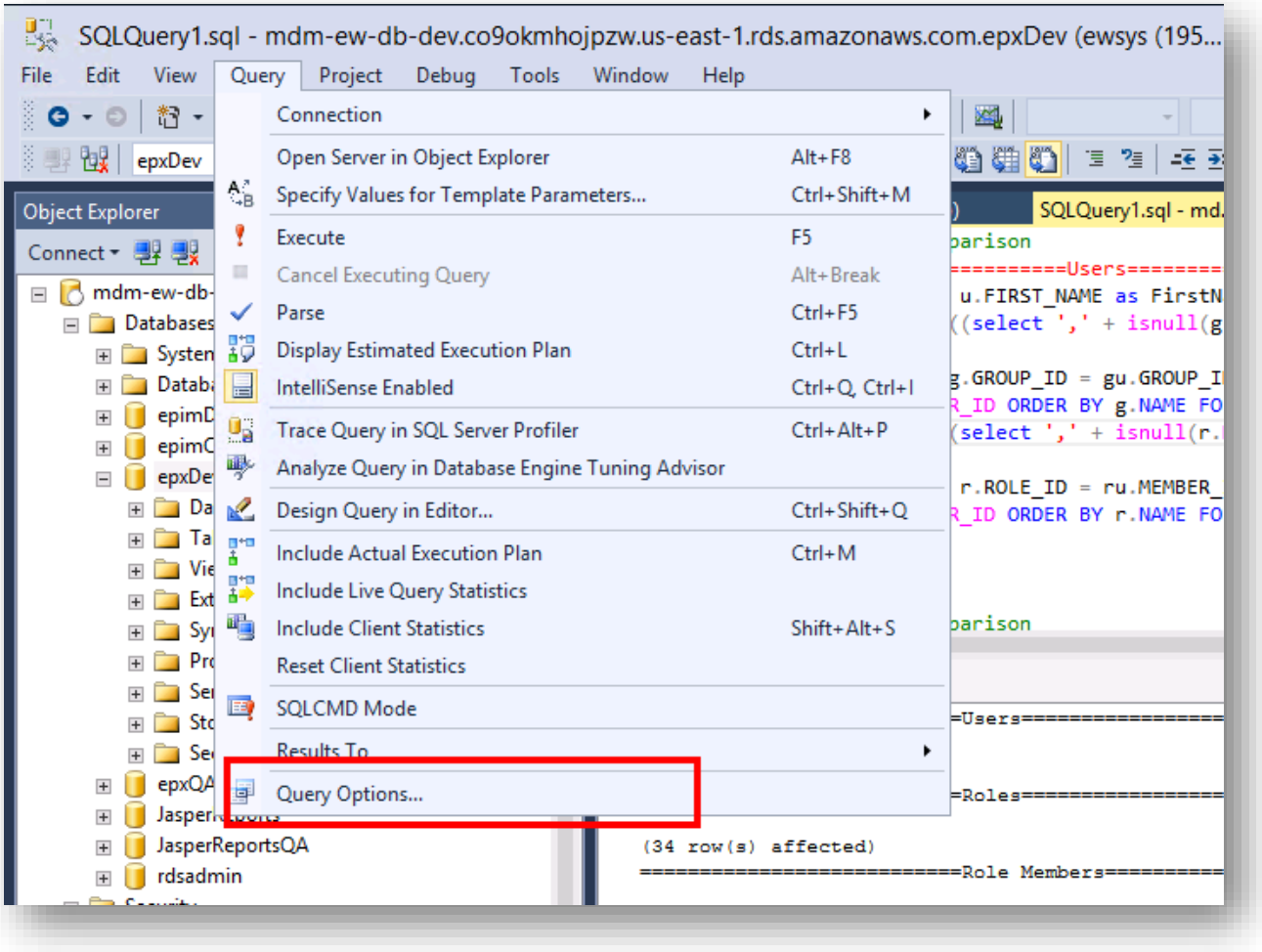
Procedure - Generate Full Report

To generate a report of all objects within an EPIM or EPX database for different environments (e.g., DEV vs. QA/PROD), perform the following steps:

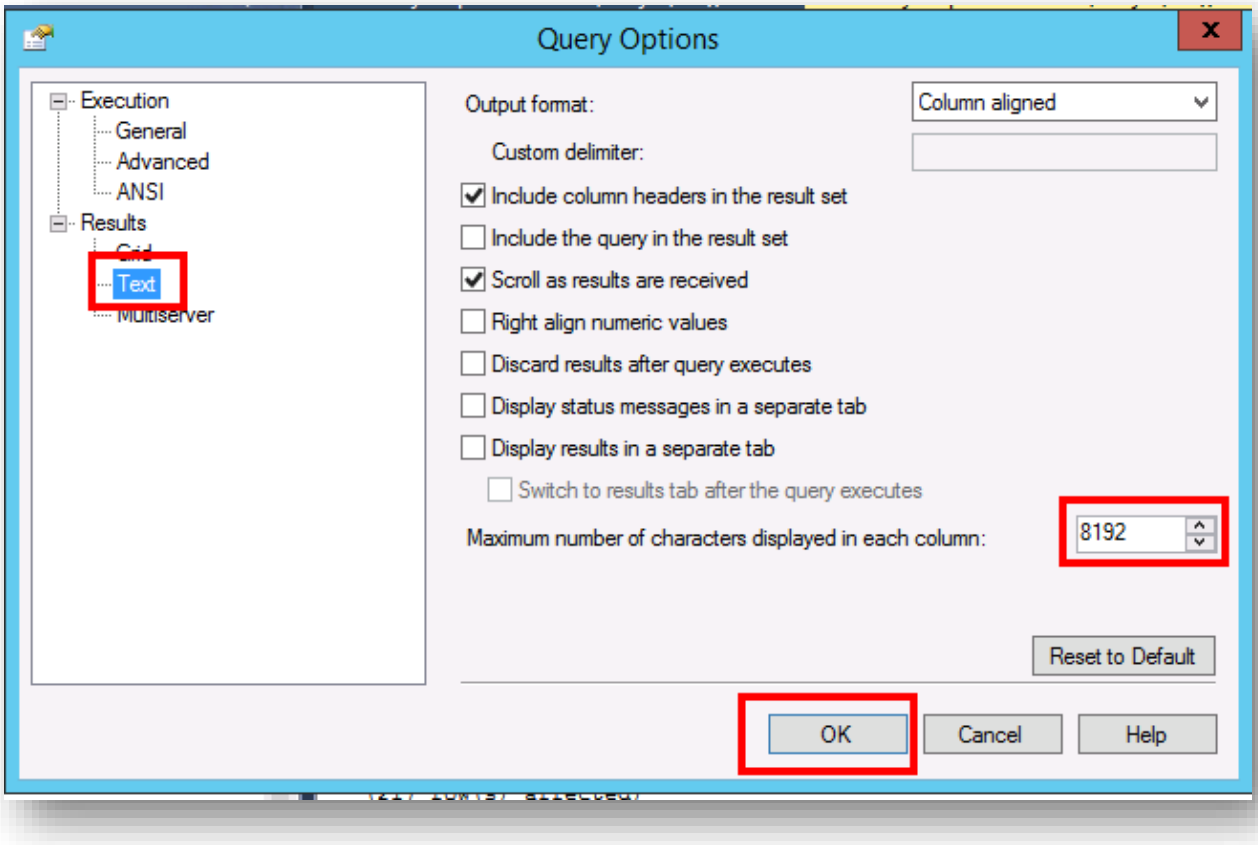
1. Connect to the EPIM or EPX database using the SQL Server Management Studio application.
2. Open a New Query window on the EPX database.
3. Load the desired compare extraction script into a new Query window.
4. Select the Results to File option:



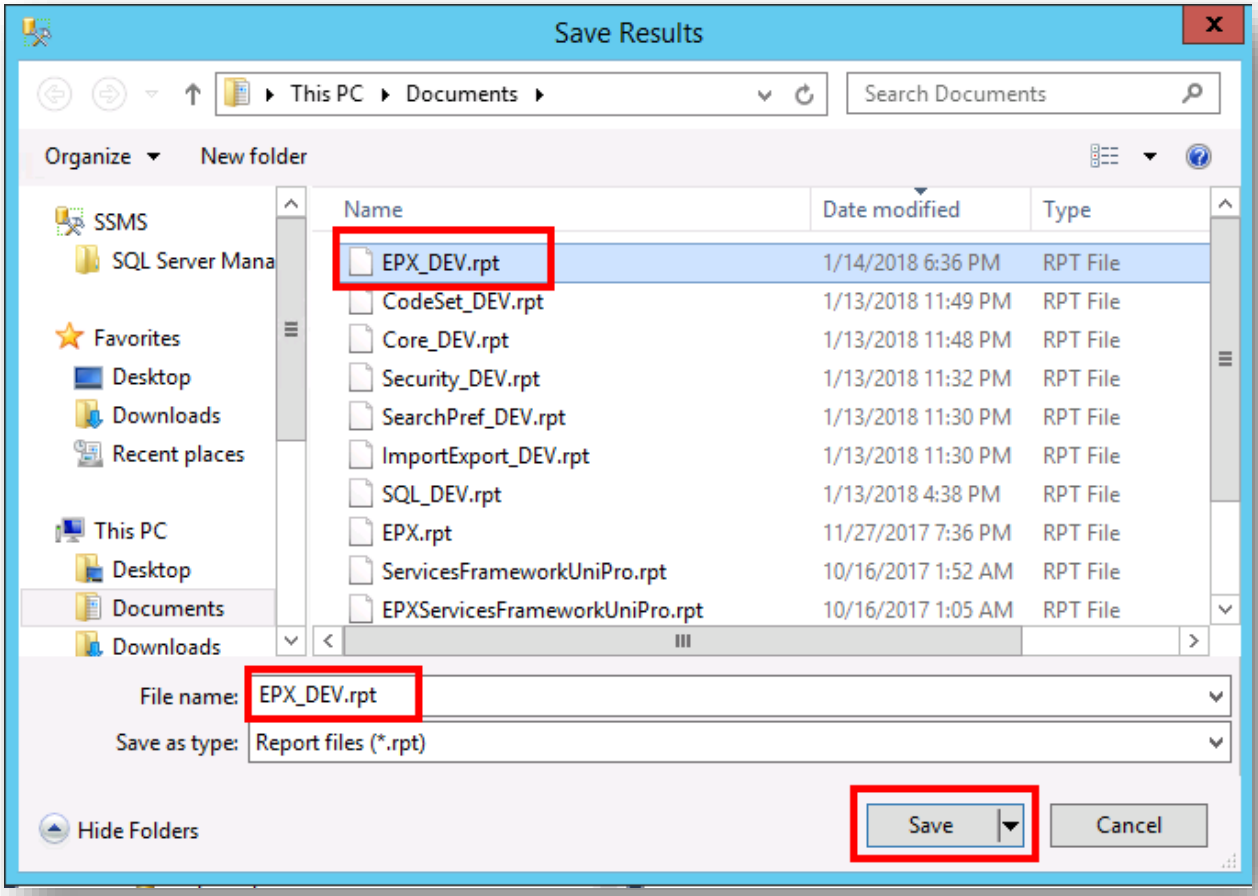
5. Select menu option Query->Query Options…:



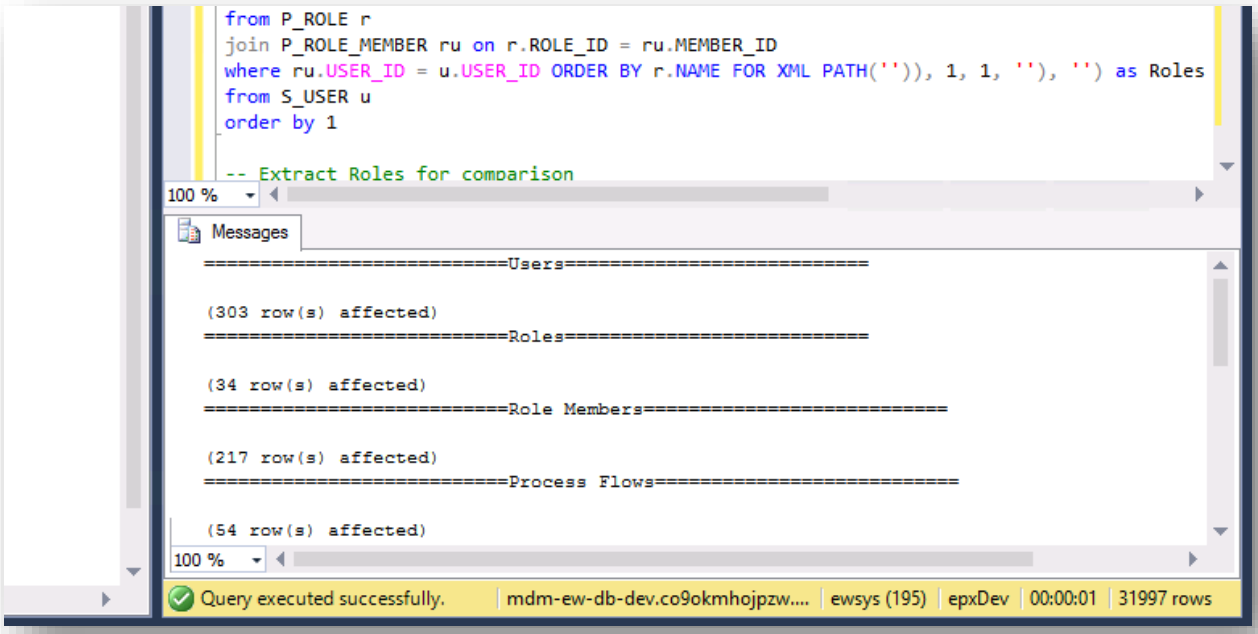
6. Set the Maximum number of characters displayed in each column to 512 or 8192 (see table for the proper size for each compare extraction script) for Results->Text and click OK:



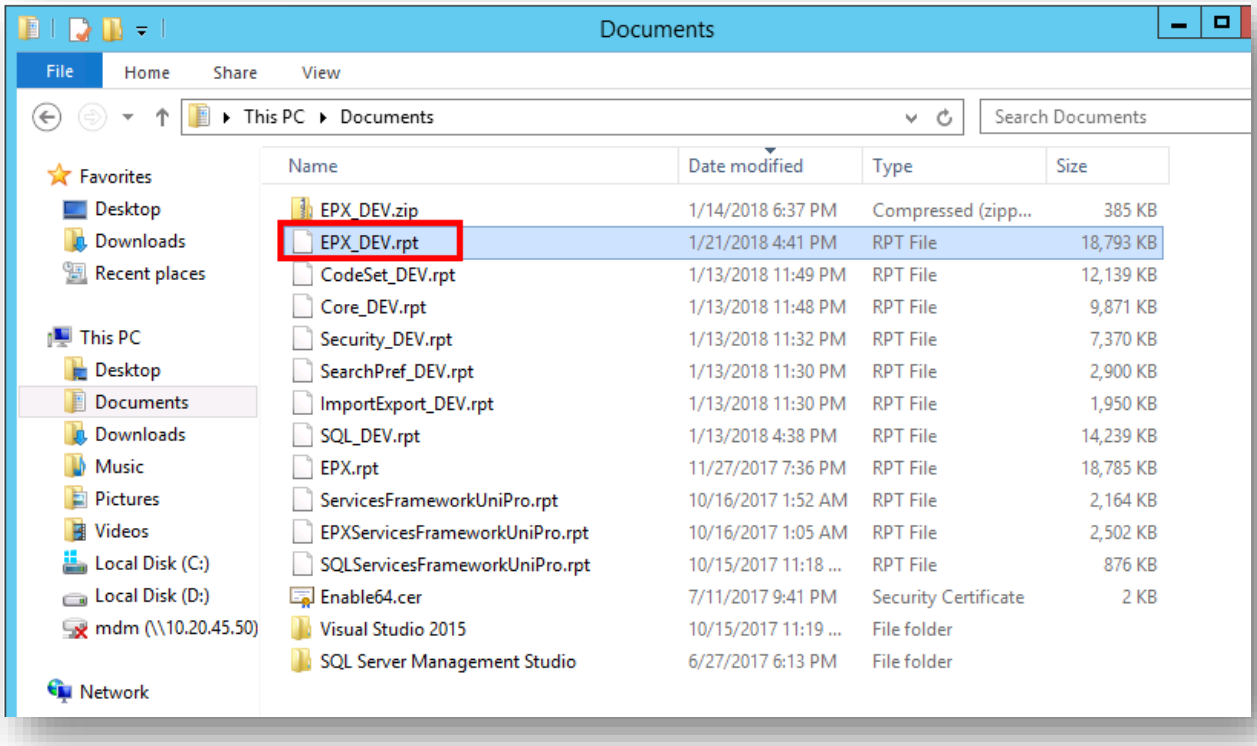
7. Execute the query. A prompt appears to select or specify the .rpt file to be created. Select or enter the name of the file and click Save:



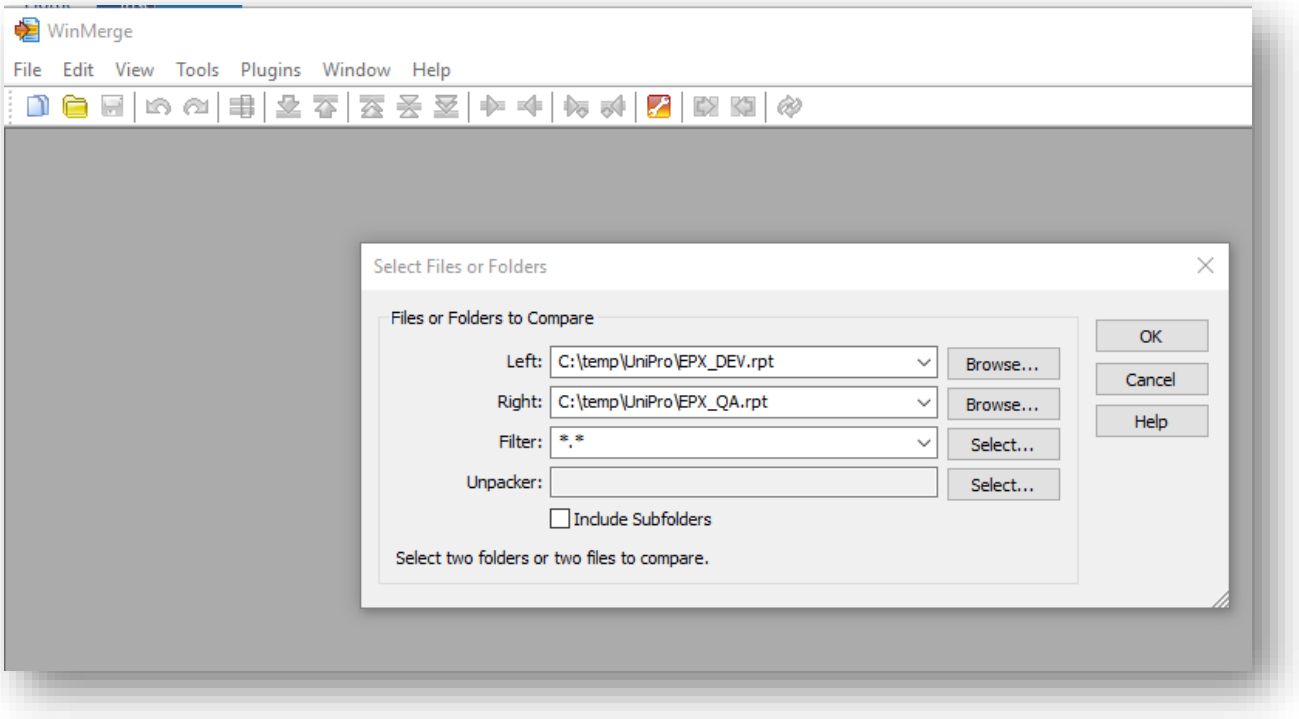
8. The Messages field will show the results of each query being executed:



9. Copy the resulting .rpt file and use in comparisons and to update CloudForge:



10. Repeat the above steps on the same database using the same script for the second environment.
11. Run the WinMerge utility to compare the two rpt files:



Procedure - Generate Partial EPX Report

An extract that includes all workflows is most useful when comparing the EPX configuration between two environments (e.g., DEV vs. QA or PROD) and the generated rpt file can be fairly large. For facilitating the archiving of workflow-specific flows in the chosen source control platform, or for migration of a subset of workflows from one environment to another, having a generated file of a subset of the EPX objects can be more useful. The partial EPX report can be generated by performing the following steps:

1. Make a copy of the CompareEPXServerExtract.sql script and edit it.
2. Eliminate or separate the following sections to different file(s):

```
-- Extract Users for comparison
print '==========================Users==========================='

-- Extract Roles for comparison
print '==========================Roles==========================='

-- Extract Role Members for comparison
print '==========================Role Members===================='
```

3. Edit each of the remaining sections to include a new condition in the WHERE clause to limit the report to a specific set of workflows:
    a. For the section:

```
-- Extract Process Flows for comparison
print '==========================Process Flows======='
```

    b. Add the condition:

```
WHERE p.NAME in ('<workflow1>', '<workflow2>', '<workflow3>')
```

    c. After:

```
FROM [P_PROCESS] p
```

    d. For the section:

```
-- Extract Process Activities for comparison
print '==========================Process Flow Activities======'
```

    e. Add the condition:

```
AND p.NAME in ('<workflow1>', '<workflow2>', '<workflow3>')
```

    f. After:

```
where a.DELETED_IND = 0
```

    g. For the section:

```
-- Extract Process Activity Viewers for comparison
print '==========================Process Flow Activity Viewers======='
```

    h. Add the condition:

```
AND p.NAME in ('<workflow1>', '<workflow2>', '<workflow3>')
```

    i. After:

```
WHERE a.DELETED_IND = 0
```

    j. For the section:

```
-- Extract Process Flow Activity Transitions for comparison
```

```
print '============================Process Flow Activity Transitions======='
```

k. Add the condition:

```
AND p.NAME in ('<workflow1>', '<workflow2>', '<workflow3>')
```

l. After:

```
AND a2.DELETED_IND = 0
```

4. In each case, the condition IN clause should list the names of the process flows or subflows to be included in the report. For example, if a report is needed on the New Brand Approval workflow, the altered file would be:

```
-- Extract Process Flows for comparison
print '============================Process Flows============================='
SELECT 'Flow=' + p.[NAME] as ProcessFlowName
, 'Desc=' + isnull(p.DESCRIPTION, '') as ProcessFlowDescription
,case when (p.[PROCESS_TYPE_CODE] = 1) then 'Process Flow'
       when (p.[PROCESS_TYPE_CODE] = 2) then 'SubFlow'
       when (p.[PROCESS_TYPE_CODE] = 3) then 'Personal SubFlow'
       else CAST(p.[PROCESS_TYPE_CODE] as VARCHAR) end as FlowType
      ,'FlowValid=' + case when (p.[VALID_IND] = 1) then 'Yes' else 'No' end as
Valid
  FROM [P_PROCESS] p
  WHERE p.NAME in ('New Brand Approval')
  order by p.[NAME]

-- Extract Process Activities for comparison
print '============================Process Flow
Activities============================='
SELECT '**** Flow=' + p.[NAME] as ProcessFlowName
      , 'Activity=' + a.NAME as ActivityName
      , 'Type=' + case when (a.ACTIVITY_TYPE_CODE = 1) then 'AUTOMATIC: ' +
a.ARC_ACTOR_NAME
    when (a.ACTIVITY_TYPE_CODE = 2) then 'SUBFLOW'
    when (a.ACTIVITY_TYPE_CODE = 3) then 'DECISION_POINT'
    when (a.ACTIVITY_TYPE_CODE = 4) then 'DISTRIBUTED_SUBFLOW'
    when (a.ACTIVITY_TYPE_CODE = 5) then 'WORK_ITEM_MERGE'
    when (a.ACTIVITY_TYPE_CODE = 6) then 'ITERATION'
    when (a.ACTIVITY_TYPE_CODE = 7) then 'JOIN: ' + case when (a.JOIN_TYPE = 1)
then 'OR' else 'AND' end
    when (a.ACTIVITY_TYPE_CODE = 8) then 'MANUAL'
    when (a.ACTIVITY_TYPE_CODE = 9) then 'ANONYMOUS'
    when (a.ACTIVITY_TYPE_CODE = 10) then 'PERSONAL_SUBFLOW'
    when (a.ACTIVITY_TYPE_CODE = 11) then 'SPLIT'
    when (a.ACTIVITY_TYPE_CODE = 12) then 'WORK_ITEM_REPEATER'
    when (a.ACTIVITY_TYPE_CODE = 14) then 'WORK_ITEM_PURGE'
    when (a.ACTIVITY_TYPE_CODE = 16) then 'ENDING_POINT'
    when (a.ACTIVITY_TYPE_CODE = 17) then 'LOAD_BALANCE'
    when (a.ACTIVITY_TYPE_CODE = 18) then 'CHANGE_PRIORITY'
```

```
    when (a.ACTIVITY_TYPE_CODE = 19) then 'SUBFLOW_EXIT'
    end as ActivityType
    , 'Enabled=' + case when (a.enabled_ind = 1) then 'E' else 'D' end as Enabled
    , 'Start=' + case when (a.start_point_ind = 1) then 'Y' else 'N' end as Start
    , 'End=' + case when (a.end_point_ind = 1) then 'Y' else 'N' end as [End]
    , 'SendOnError=' + case when (a.ERROR_SEND_IND = 1) then 'Y' else 'N' end as
[SendOnError]
    , 'Valid=' + case when (a.VALID_IND = 1) then 'Y' else 'N' end as
ActivityValid
    , 'Key=' + ap.PROPERTY_KEY + ' = ' +
        isnull(case when (ap.PROPERTY_KEY = 'lastSentDate') THEN ''
        else case when (ap.PROPERTY_VALUE is not null) then ap.PROPERTY_VALUE
                else isnull(convert(varchar(max),
convert(varbinary(max),ap.PROPERTY_VALUE_BLOB)),'')
                end
        end, '') as ActivityPropertyKey
  FROM [P_PROCESS] p
  join P_ACTIVITY a on p.PROCESS_ID = a.PROCESS_ID
  left outer join P_ACTIVITY_PROPERTY ap on a.ACTIVITY_ID = ap.ACTIVITY_ID
  where a.DELETED_IND = 0
  AND p.NAME in ('New Brand Approval')
  order by p.[NAME], a.NAME, ap.PROPERTY_KEY


-- Extract Process Activity Viewers for comparison
print '==========================Process Flow Activity
Viewers=========================='
SELECT '**** Flow=' + p.[NAME] as ProcessFlowName
      , 'Activity=' + a.NAME as ActivityName
      , 'Start=' + case when (a.start_point_ind = 1) then 'Y' else 'N' end as
Start
      , 'Viewer=' + wiv.NAME as ViewerName
      , 'URL=' + wiv.URL as ViewerURL
      , 'Default=' + case when wiv.DEFAULT_IND = 0 then 'No'
                when wiv.DEFAULT_IND = 1 then 'Yes'
                else 'Unknown:' + CAST(wiv.default_ind as VARCHAR)
                end as DefaultViewer
  FROM [P_PROCESS] p
  join P_ACTIVITY a on p.PROCESS_ID = a.PROCESS_ID
  join p_ACTIVITY_VIEWER av on av.ACTIVITY_ID = a.ACTIVITY_ID
  join P_WORK_ITEM_VIEWER wiv on av.VIEWER_ID = wiv.VIEWER_ID
  WHERE a.DELETED_IND = 0
  AND p.NAME in ('New Brand Approval')
  order by p.[NAME], a.NAME, wiv.NAME


-- Extract Process Flow Activity Transitions for comparison
print '==========================Process Flow Activity
Transitions=========================='
SELECT 'Flow=' + p.[NAME] as ProcessFlowName
      , 'From=' + a.NAME as ActivityName
      , 'To=' + isnull(a2.NAME, '') as TargetActivityName
      , case when (tc.CONDITION_ID IS not null) then
      'Con=' + case when (c.CONDITION_TYPE_CODE = 3) then 'Otherwise'
```

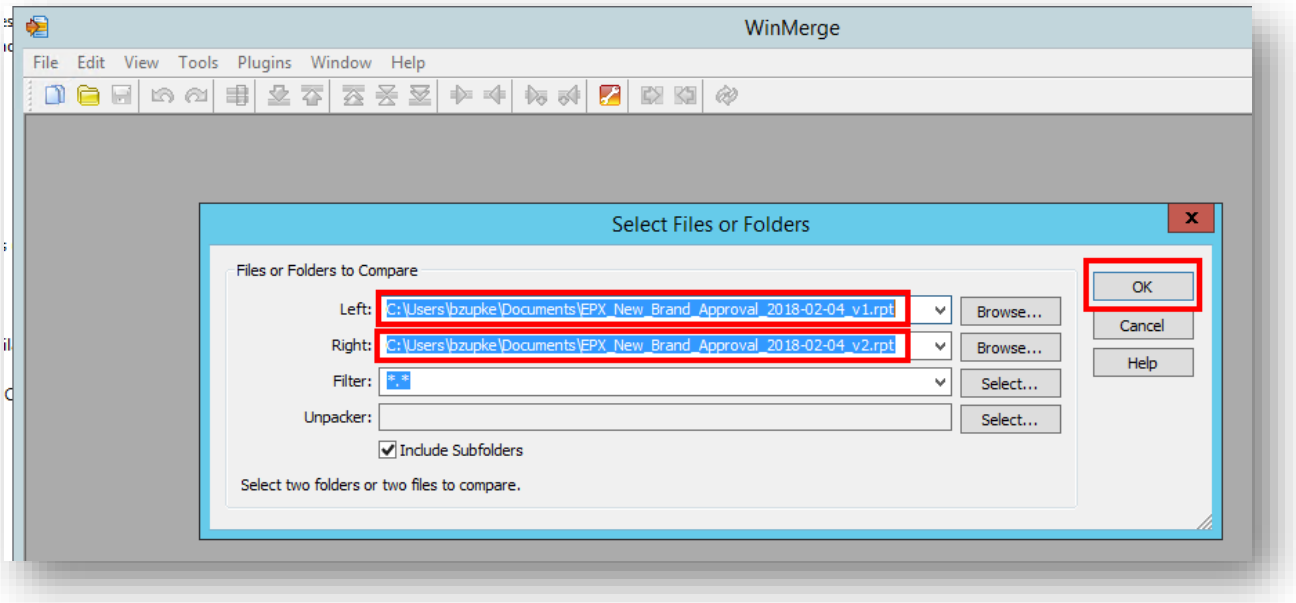
```
      when (c.CONDITION_TYPE_CODE = 1) then 'Simple: ' + c.COMPARE_KEY +
C.OPERATOR_CODE + c.COMPARE_VALUE
      when (c.CONDITION_TYPE_CODE = 2) then 'Advanced: ' +
isnull(convert(nvarchar(max),c.FREE_EXPRESSION_CLOB), '')
      else 'Unknown: ' + cast(c.CONDITION_TYPE_CODE as varchar) end
      else '' end as Condition
  FROM [P_PROCESS] p
  join P_ACTIVITY a on p.PROCESS_ID = a.PROCESS_ID
  left outer join P_TRANSITION t on t.PARENT_ACTIVITY_ID = a.ACTIVITY_ID
  left outer join P_ACTIVITY a2 on t.CHILD_ACTIVITY_ID = a2.ACTIVITY_ID
  left outer join P_TRANSITION_CONDITION tc on tc.TRANSITION_ID = t.TRANSITION_ID
  left outer join P_CONDITION c on tc.CONDITION_ID = c.CONDITION_ID
  WHERE a.DELETED_IND = 0
  AND a2.DELETED_IND = 0
  AND p.NAME in ('New Brand Approval')
  order by 1,2,3,4
```

5. Using the modified script, follow ***Procedure - Generate Full Report***
6. Reports on individual process flows or subflows can be archived in the source control repository as separate artifacts. The native comparison tool or an external comparison tool such as WinMerge can be used to compare two versions of the same workflow.

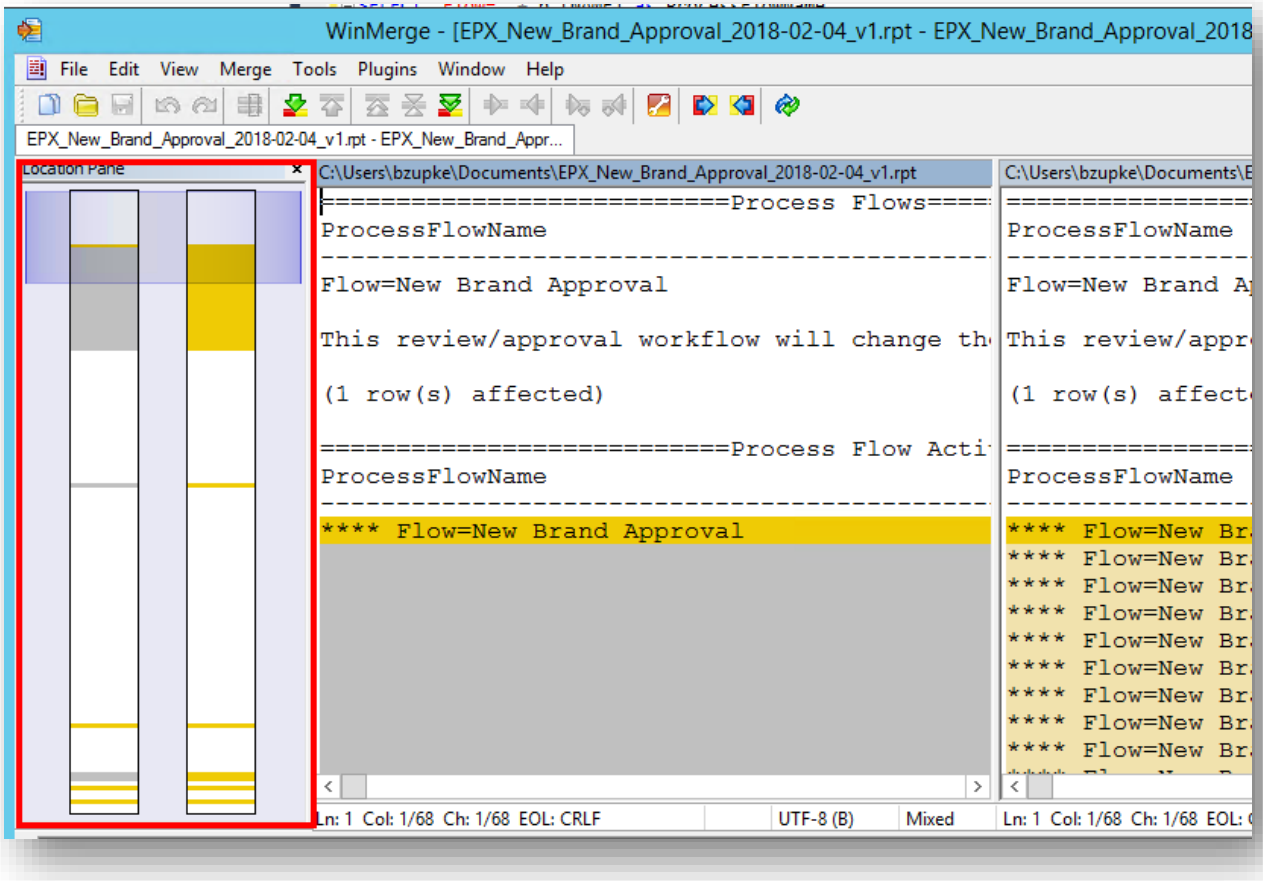
Procedure – Comparing Extraction Reports Using WinMerge

Two reports generated using the same extraction SQL can be compared using WinMerge, which will highlight the differences between the two versions by performing the following steps:

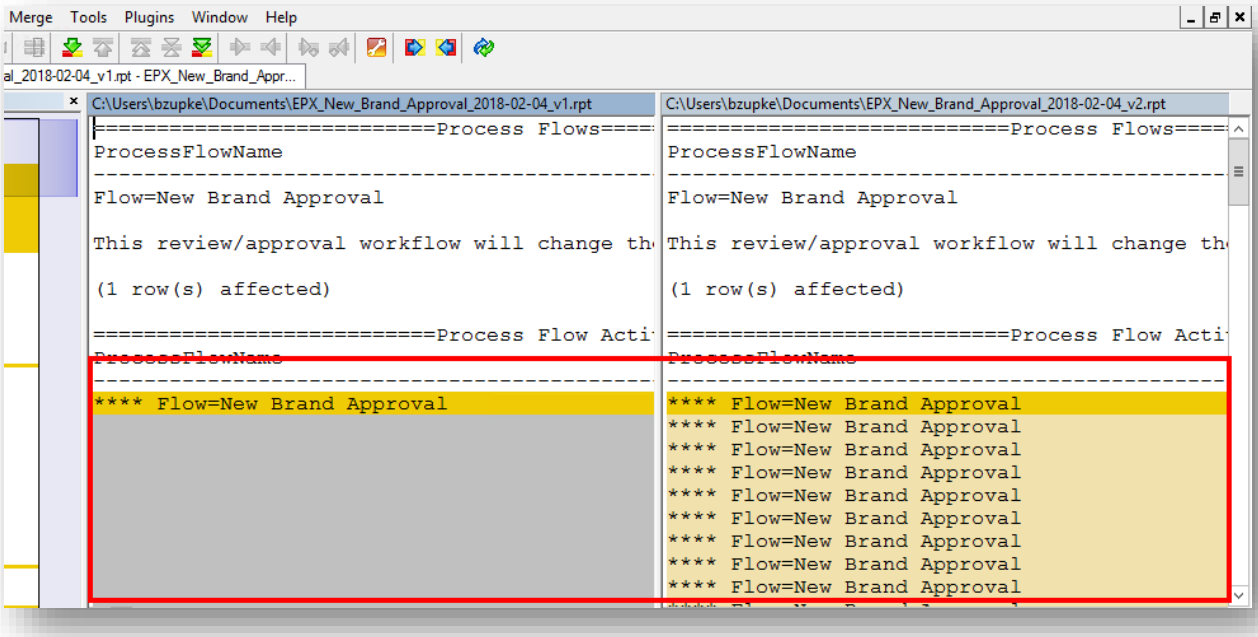
1. Open WinMerge.
2. Select two versions of the report file generated by the same query in either two different systems or from the same system before and after changes are made:



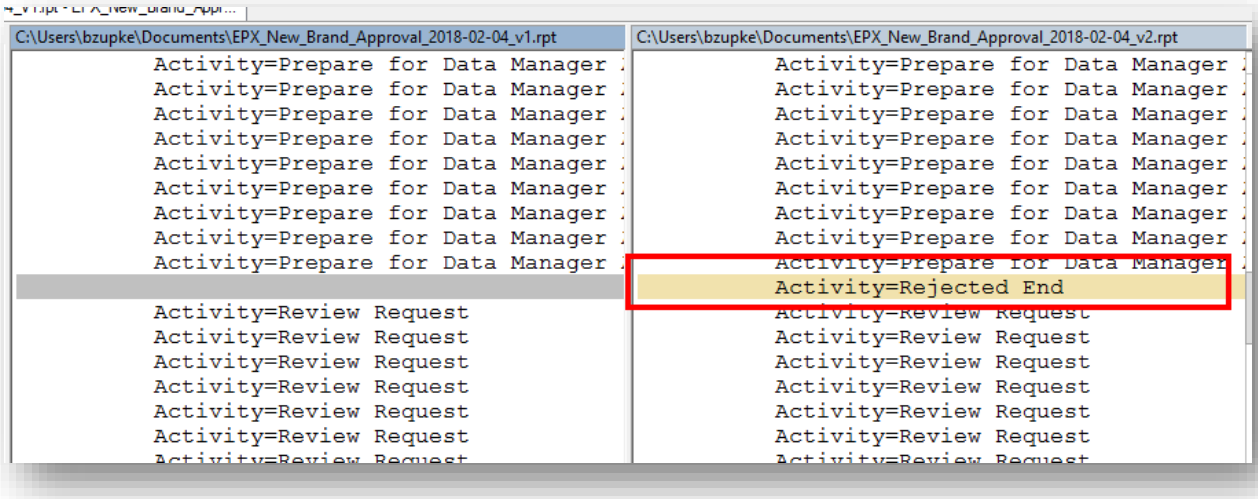
3. The tool graphically highlights the sections of the two files that have differences on the left:



4. The tool graphically highlights the lines in the files that are different on the right:



5. By scrolling up and down, the details of those differences can be seen. For example, the files being compared show that the first version did not have the Rejected End activity:



Deployment Script Examples

The following scripts are examples of deployment scripts that can be used as a starting point.

In simple environments where it will likely be necessary to deploy binaries (which require services restarts) less frequently than non-binaries. In these such environments, a simple set of scripts can be defined:

deploy.bat – calls the non-binaries scripts and deploys the binaries and runs the necessary configuration scripts:

```
copy Greenheck.jar F:\Enterworks\lib\Services\lib
copy Greenheck.jar F:\Enterworks\EnableServer\tomcat\webapps\webcm\WEB-INF\lib

call deployFilesOnly.bat

REM Load SQL Scripts
ECHO update sQL from DEV
pause
REM call deploySqlOnly.bat

call F:\Enterworks\bin\DeployServicesJar.bat
pause
```

It's a good practice to end the scripts with a pause command so the command prompt window does not close until you have had a chance to review the output status of each command.

deployFilesOnly.bat – copies the non-binary files and can be run without having to restart services:

```
copy *.jsp F:\Enterworks\EnableServer\tomcat\webapps\webcm\custom

copy coreLanguageScripts\*.*
F:\Enterworks\EnableServer\tomcat\webapps\webcm\shared\core\scripts\i18n
copy core2LanguageScripts\*.*
F:\Enterworks\EnableServer\tomcat\webapps\webcm\shared\core2\scripts\i18n
copy channelReadinessLanguageScripts\*.* F:\Enterworks\enable-mean\angular-
ui\ng\channel-readiness\languages
pause
```

deploySqlFilesOnly.bat – invokes the SQL Server commands to load any SQL scripts into the EPIM database. This script must be run on a server that has the SQL Server Management Studio client installed:

```
REM Load SQL Scripts
sqlcmd -S dbServerName -d EPIM -U epimsys -P password -i project_script1.sql -o
project_script1.out

sqlcmd -S dbServerName -d EPIM -U epimsys -P password -i project_script2.sql -o
project_script2.out

sqlcmd -S dbServerName -d EPIM -U epimsys -P password -i project_script3.sql -o
project_script3.out

sqlcmd -S dbServerName -d EPIM -U epimsys -P password -i project_script4.sql -o
project_script4.out
```

For more complex environments where the file structure and even which components are installed are different on each target (e.g., separate servers for WEB and APP), a self-identifying script can be defined.

Such scripts may take a while to develop, but they can be re-used from one deployment to the next and refined with each subsequent deployment:

```
REM Deploy Files for [project] Vendor Portal Update 3/4/2016
REM
REM This script copies all files needed for the [project] Vendor Portal
REM update.
REM It must be run on the server to which the files are being copied and
REM from the directory containing this script (all file references are
REM relative).  It should be run on each server comprising the environment
REM for both the Vendor Portal and the Internal PIM.
REM
REM Default to E: drive.  Set to C: drive if on local machine for testing
set DRIVE=C:
IF "%COMPUTERNAME%"=="PANICILLIN" GOTO :SKIP_EDRIVE
set DRIVE=E:
:SKIP_EDRIVE
echo Drive=%DRIVE%

REM It recognizes the target by host name.  If the host names change,
REM this script needs to be updated.  The current hosts are:
REM
REM PIMDEV2 - DEV Vendor Portal Server
REM VPWEBTEST - TEST Vendor Portal WEB Server
REM VPAPPTEST - TEST Vendor Portal APP Server
REM VPWEB - PROD Vendor Portal WEB Server
REM VPAPP - PROD Vendor Portal APP Server
REM PIMDEV - DEV Internal PIM
REM MPWEBTEST - TEST Internal PIM WEB Server
REM MPAPPTEST - TEST Internal PIM APP Server
REM MPWEB - PROD Internal PIM WEB Server
REM MPAPP - PROD Internal PIM APP Server
REM
set host=%COMPUTERNAME%
echo .%host%.


if "%host%"=="PANICILLIN" GOTO :VPDEV
if "%host%"=="PIMDEV2" GOTO :VPDEV
if "%host%"=="VPWEBTEST" GOTO :VPWEB
if "%host%"=="VPAPPTEST" GOTO :VPAPP
if "%host%"=="VPWEB" GOTO :VPWEB
if "%host%"=="VPAPP" GOTO :VPAPP
if "%host%"=="PIMDEV" GOTO :MPDEV
if "%host%"=="MPWEBTEST" GOTO :MPWEB
if "%host%"=="MPAPPTEST" GOTO :MPAPP
if "%host%"=="MPWEB" GOTO :MPWEB
if "%host%"=="MPAPP" GOTO :MPAPP

echo Host %host% not expected DEV, APP or WEB server.  Deployment aborted.
GOTO :END

REM Set the flags used to determine what files to copy/delete
```

```
:VPDEV
SET serverType=DEV
SET enableType=VP
GOTO :CopyFiles

:VPWEB
SET serverType=WEB
SET enableType=VP
GOTO :CopyFiles

:VPAPP
SET serverType=APP
SET enableType=VP
GOTO :CopyFiles

:MPDEV
SET serverType=DEV
SET enableType=MP
GOTO :CopyFiles

:MPWEB
SET serverType=WEB
SET enableType=MP
GOTO :CopyFiles

:MPAPP
SET serverType=APP
SET enableType=MP
GOTO :CopyFiles

:CopyFiles
ECHO Server Type = %serverType%

REM ************************************************************************
REM
REM         JBoss Master File Deployment
REM
REM ************************************************************************

REM Master JBOSS on DEV and WEB only
if "%serverType%"=="APP" GOTO :SkipMaster
copy [Project].jar %DRIVE%\Enterworks\EnableServer\jbossMaster\server\default\lib
copy Services.jar %DRIVE%\Enterworks\EnableServer\jbossMaster\server\default\lib

REM Remove Obsolete files
DEL %DRIVE%\Enterworks\EnableServer\jbossMaster\server\default\lib\groovy-all-
2.3.7.jar


:SkipMaster
REM ************************************************************************
REM
REM         JBoss Slave1, Slave2 File Deployment
```

```
REM
REM ************************************************************************

REM Slave 1&2 ON DEV and APP only
if "%serverType%"=="WEB" GOTO :SkipSlave1And2
copy [project].jar %DRIVE%\Enterworks\EnableServer\jbossSlave1\server\default\lib
copy [Project].jar %DRIVE%\Enterworks\EnableServer\jbossSlave2\server\default\lib
copy Services.jar %DRIVE%\Enterworks\EnableServer\jbossSlave1\server\default\lib
copy Services.jar %DRIVE%\Enterworks\EnableServer\jbossSlave2\server\default\lib

if "%enableType%"=="MP" GOTO :MPCopyMigration

copy VendorPortalMigration.msf %DRIVE%\Enterworks\shared\migration
copy VendorPortalMigration.zip %DRIVE%\Enterworks\shared\migration
GOTO :EndCopyMigration

:MPCopyMigration
copy InternalPIMMigration.msf %DRIVE%\Enterworks\shared\migration
copy InternalPIMMigration.zip %DRIVE%\Enterworks\shared\migration


:EndCopyMigration

REM Remove Obsolete files
DEL %DRIVE%\Enterworks\EnableServer\jbossSlave1\server\default\lib\groovy-all-
2.3.7.jar
DEL %DRIVE%\Enterworks\EnableServer\jbossSlave2\server\default\lib\groovy-all-
2.3.7.jar


:SkipSlave1And2
REM ************************************************************************
REM
REM         JBoss Slave3 File Deployment
REM
REM ************************************************************************

REM Slave 3 ON APP only
if "%serverType%"=="DEV" GOTO :SkipSlave3
if "%serverType%"=="WEB" GOTO :SkipSlave3
copy [Project].jar %DRIVE%\Enterworks\EnableServer\jbossSlave3\server\default\lib
copy Services.jar %DRIVE%\Enterworks\EnableServer\jbossSlave3\server\default\lib

REM Remove Obsolete files
DEL %DRIVE%\Enterworks\EnableServer\jbossSlave3\server\default\lib\groovy-all-
2.3.7.jar


:SkipSlave3
REM ************************************************************************
REM
REM         JBoss Slave4 File Deployment
REM
REM ************************************************************************
```

```
REM Slave 4 ON WEB only
if "%serverType%"=="DEV" GOTO :SkipSlave4
if "%serverType%"=="APP" GOTO :SkipSlave4
copy [Project].jar %DRIVE%\Enterworks\EnableServer\jbossSlave4\server\default\lib
copy Services.jar %DRIVE%\Enterworks\EnableServer\jbossSlave4\server\default\lib

REM Remove Obsolete files
DEL %DRIVE%\Enterworks\EnableServer\jbossSlave4\server\default\lib\groovy-all-
2.3.7.jar


:SkipSlave4
REM ******************************************************************************
REM
REM         Tomcat File Deployment
REM
REM ******************************************************************************

REM Tomcat ON DEV and WEB only
if "%serverType%"=="APP" GOTO :SkipTomcat
copy *.jsp %DRIVE%\Enterworks\EnableServer\tomcat\webapps\webcm\custom
copy [Project].jar %DRIVE%\Enterworks\EnableServer\tomcat\webapps\webcm\WEB-
INF\lib
copy Services.jar %DRIVE%\Enterworks\EnableServer\tomcat\webapps\webcm\WEB-INF\lib

REM Remove Obsolete files
DEL %DRIVE%\Enterworks\EnableServer\tomcat\webapps\webcm\WEB-INF\lib\groovy-all-
2.3.7.jar

:SkipTomcat
REM ******************************************************************************
REM
REM         EPX File Deployment
REM
REM ******************************************************************************

REM EPX on DEV and APP only
if "%serverType%"=="WEB" GOTO :SkipEPX

REM Remove Obsolete files

del %DRIVE%\Enterworks\EPX\bin\agents\%host%\FtpBIC.jar
del %DRIVE%\Enterworks\EPX\bin\agents\client\EPX_%host%\FtpBIC.jar
DEL %DRIVE%\Enterworks\EPX\lib\groovy-all-2.3.7.jar

REM Copy Files

copy [Project].jar %DRIVE%\Enterworks\EPX\lib
copy Services.jar %DRIVE%\Enterworks\EPX\lib
copy FtpBic.config %DRIVE%\Enterworks\EPX\bin\agents\%host%
copy FTPBicGroovy.jar %DRIVE%\Enterworks\EPX\bin\agents\%host%
copy *.groovy %DRIVE%\Enterworks\EPX\groovy\bic
```

```
REM Create new directories


if "%enableType%"=="MP" GOTO :SKIP_VP_DIRECTORIES

IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles GOTO :SKIP_VENDOR_FILES
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles
:SKIP_VENDOR_FILES

IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles\Drop GOTO
:SKIP_VENDOR_FILES_DROP
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles\Drop
:SKIP_VENDOR_FILES_DROP

IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles\epaCUBE GOTO :SKIP_EPACUBE
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles\epaCUBE
:SKIP_EPACUBE


IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles\Invalid GOTO
:SKIP_VENDOR_FILES_INVALID
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles\Invalid
:SKIP_VENDOR_FILES_INVALID

IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles\MultiVendorDrop GOTO
:SKIP_VENDOR_FILES_MULTI_VENDOR_DROP
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles\MultiVendorDrop
:SKIP_VENDOR_FILES_MULTI_VENDOR_DROP

IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles\Reports GOTO
:SKIP_VENDOR_FILES_REPORTS
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles\Reports
:SKIP_VENDOR_FILES_REPORTS

IF EXIST %DRIVE%\Enterworks\[Project]\VendorFiles\Staging GOTO
:SKIP_VENDOR_FILES_STAGING
mkdir %DRIVE%\Enterworks\[Project]\VendorFiles\Staging
:SKIP_VENDOR_FILES_STAGING

:SKIP_VP_DIRECTORIES

:SkipEPX
Pause
```