Tivoli® software

IBM

Migration Use Cases with the Migration Manager Version 7.5

Learn about Migration Manager implementation strategy

Experiment with various real-life scenarios

Review migration troubleshooting tips

Rebecca Alsop
Mohamed Amine Bourenane
Brian Davis
Bradley Downing
Vasfi Gucer
Armen Pischdotchian
Sampath Sriramadhesikan

Redbooks

ibm.com/redbooks

International Technical Support Organization

Migration Use Cases with the Migration Manager Version 7.5

March 2013

SG24-8132-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

First Edition (March 2013)

This edition applies to Tivoli process automation engine V7.5 and IBM SmartCloud Control Desk V7.5

Contents

Notices

Trademarks

Preface

By using the Migration Manager, you can migrate configuration content from one production environment to another. The typical use is to migrate configuration content from a development environment to a test environment and then on to production for the Tivoli® process automation engine and its applications, such as IBM® SmartCloud® Control Desk. The goal of migration is to ensure that your production environment fully meets the needs of your users.

This IBM Redbooks® publication is an update of the existing book *Migration Use Cases with the Migration Manager,* SG24-7906 and covers the most common migration use cases with the Migration Manager, including the capabilities that were introduced with Tivoli's process automation engine V7.5. These use cases are only a small subset of the possible migration scenarios that can be performed by the Migration Manager, but they were chosen to be representative of the capabilities of the Migration Manager.

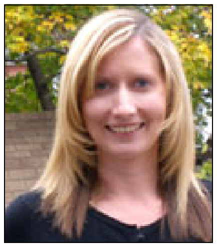
In addition to these use cases, the book presents a migration strategy and a comprehensive chapter about troubleshooting possible migration problems when the Migration Manager is used.

We strongly suggest that you read Chapter 1, "Migration strategy" on page 1 first before reading the other chapters. This chapter provides a good foundation for all of the migration scenarios that are covered in the book.

This book is a reference for IT Specialists and IT Architects working on migrating configuration content from one production environment to another by using the Migration Manager.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Rebecca Alsop is a Senior Consultant with the Createch Group, a Bell Canada Company. She has six years of experience working with Maximo in various industries including telecommunications, government, and transportation. She is involved in many Maximo integration and customization projects and has participated in all components of technical consulting, development, and implementation. She is ITIL certified and has a BS Honors Degree in Computer Science from McMaster University.

Mohamed Amine Bourenane is an IBM Maximo® Consultant with the Createch Group, a Bell Canada company. He began his professional carrier 13 years ago in oil and gas industry working for JGC Corporation and British Petroleum. Amine is a Maximo Certified Deployment Professional who specializes in Maximo implementation, customization, and integration for telecommunications, construction, electricity, and manufacturing companies. He holds a Masters degree in Computer Science from University of Quebec.

Brian Davis works as the technical team leader in the Tivoli Support organization. He has spent the last three years working on Tivoli IBM Service Request Manager®, Tivoli Asset Manager and CCMDB deployments. His specialty is in the creation of integrations and the use of Migration Manager. He was one of the authors of the second edition of *Identity Management Design Guide with IBM Tivoli Identity Manager*, SG24-6996-01 IBM Redbooks.

Bradley Downing works for IBM in the Tivoli Advanced Technology Group - ISM Lab Services. He began his career with the Maximo product 15 years ago with a small integrator in California. In 2001, he joined MRO Software and came to IBM through an acquisition in 2006. During that time, he has specialized in working with clients in various industries, with emphasis on reporting, Mobile Maximo, the use of the Integration Framework, and recently with the Migration Manager module. He holds an MBA in Business Management.

Vasfi Gucer is a Project Leader at the International Technical Support Organization, Austin Center. He has been with the ITSO since January 1999. He has over 15 years of experience in the areas of systems management, networking hardware, and software on mainframe and distributed platforms. He worked on various IBM Tivoli client projects as a Systems Architect in the US. He writes extensively and teaches IBM classes worldwide on Tivoli software. Vasfi is also an IBM Certified Senior IT Specialist, PMP, and ITIL Expert.

Armen Pischdotchian joined IBM shortly after MRO acquisition, where he started his career as an Information Developer leading to a role in Support L2 as systems installer. He is a course developer and an instructor for Tivoli's Maximo products. His specialty is in system-related disciplines, such as Migration manager, Integration Framework, and Automation Scripting. He also develops and delivers courses in Maximo Spatial and Maximo for Service Providers. Recently, he was involved with creating and delivering a 10-day Maximo Implementation bootcamp for Business Partner Ecosystem initiative. For the past six years as an Enablement specialist, he conducted many seminars and labs at European Sales Academies and world-wide client and business partner enablement efforts. He holds two Masters degrees: Technical and Professional Writing and an MBA in Computer Information Systems. He also is an IBM Certified Fundamentals of Applying Maximo Enterprise Asset Management Solutions.

Sampath Sriramadhesikan is a lead architect in the Maximo and SmartCloud Control Desk product development organization, which focuses on configuration and customization technologies and tooling. Sampath has worked with many Smarter Physical Infrastructure (SPI) clients sharing his expertise and supporting complex implementations that required in depth configuration and customization. He has extensive experience in designing business object and integration frameworks and programming in object oriented languages.

Thanks to the following people for their contributions to this project:

- Tamikia Barrow, David Bennin, Shari Deiana

  International Technical Support Organization

- Chris Doan, Bob Dunyak, Stephen Ridgill

  IBM USA

Now you can become a published author, too!

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at this website:

  `http://www.ibm.com/redbooks`

- Send your comments in an email to:

  `redbooks@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  `http://www.facebook.com/IBMRedbooks`
- Follow us on Twitter:

  `http://twitter.com/ibmredbooks`
- Look for us on LinkedIn:

  `http://www.linkedin.com/groups?home=&gid=2130806`
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  `https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm`
- Stay current on recent Redbooks publications with RSS Feeds:

  `http://www.redbooks.ibm.com/rss.html`

1

Migration strategy

In this chapter, we show how the Migration Manager works together with other tools to successfully migrate Tivoli system configurations across the client's environments. Typically, these environments are represented as development (unit testing), test (integration testing), stage (final integration and performance testing), and production.

The Migration Manager is a component of Tivoli's process automation engine. It is a set of tools that are used to promote system configurations from a development environment to upper environments, such as stage, user acceptance testing, and production.

System configuration is a set of metadata that enables application functionality and controls application behavior in a production environment. For example, the user interface for the Service Request application is enabled and rendered through application presentation metadata.

The Migration Manager is heavily used in migrating system configurations before the production environment goes live. However, it is also used in migrating changes to system configurations after the production environment is live. We describe the role that the Migration Manager plays in both scenarios.

A production environment that produces system configurations to be migrated is often called a *source environment*. A production environment that uses system configurations is often called a *target environment*.

This chapter includes the following sections:

- Requirements
- Choice of tools to support migration
- Content strategy
- Migration approach
- Migration planning
- Migration Manager reference material

1.1 Requirements

Clients often have the following requirements for the migration of system configurations from development to production:

- Migration must be repeatable and efficient across environments that constitute the system landscape.
- Migration must be durable in that it can be used before the production environment goes live and after the production environment goes live.
- Migration must be stable in that all types of system configuration can be identified, collected, packaged, distributed, and deployed in the same manner.
- Migration must leave the target environment usable and always intact. The integrity of the system configurations must not be compromised.
- Migration must minimize or eliminate the system downtime of the target environments.

Test data, such as data that is used to drive unit testing in a development environment, is not to be promoted. Source code in the form of Java sources or compiled Java class files can be managed separately by using traditional source control systems.

1.2 Choice of tools to support migration

To meet these requirements, the delivery architect must evaluate how Tivoli is intended to operate within the enterprise. Tivoli exchanges data records with the enterprise. It is necessary to define how this data is maintained during migration and post-production, which affects the way that systems are built from one production environment to the next. For example, person records might never be migrated. Instead, each production environment synchronizes person records from a directory. Alternatively, person records might be always loaded into a production environment from an existing system, which impacts the migration plan, scope, and effort.

From a tools perspective, the Migration Manager is primarily responsible for the migration of system configuration metadata that is generated as a result of configuration and customization activity in the development environment. The Migration Manager also can migrate application content.

However, the following criteria must be applied to determine whether such application content must be migrated by using the Migration Manager:

- Is the application static? Static data never changes. It is set up once and then is used in the same static form across all production environments.
- Will the application content be created in a development environment? Application content that is created in a development environment might need to be promoted. For example, a hierarchy of classifications might support the Job Plan application.
- Can the application content be characterized as master or transactional data? Examples of master data include person, item master, and company records. Examples of transactional data include service requests, incidents, purchase orders, and invoices. These types of data are not set up in a development environment except as test data.

If application content is static, created in a development environment, and not master or transactional data, this content is a candidate to migrate to target environments that use the Migration Manager.

1.2.1 Data loading tools

From the point of view of preparing target environments, such as test or production, through data loads, a number of IBM tools exist in addition to the Migration Manager. These data loading tools are geared to deliver the following specific data loading capabilities:

- Tivoli's process automation engine Integration Framework
- Tivoli ISM Content Installer
- Tivoli Integration Composer
- Tivoli Directory Integrator

In certain cases, a business application might offer an application-specific data loading capability. For example, the Application Designer provides an export and import feature that supports application and system presentations.

The choice of tool depends on the specific data loading requirement. Table 1-1 lists particular tools and their use.

Table 1-1 Tools and use cases

Type of data loading	Tivoli's process automation engine Migration Manager	Tivoli's process automation engine Integration Framework	Tivoli ISM Content Installer	Tivoli Integration Composer	Tivoli Directory Integrator
Batch loading of data from discovery tools	No	No	No	Yes	No
Batch loading of master and transactional data or existing data	No	Yes	No	No	No
Messaging-based integration with external systems	No	Yes	No	No	Yes
Import of IBM supplied content	No	No	Yes	No	No
Promotion of system configurations between Tivoli’s process automation engine-based production environments	Yes	No	No	No	No

Based on the tools and their uses, it is clear that the Migration Manager is not the only tool that is used to load data into upper environments. Selecting the appropriate tool that is based on your data loading needs helps ensure success in meeting production go-live time frames.

For a more information about data loading tools and capabilities, see the following *Data Integration Best Practices* publication at this website:

http://www.ibm.com/developerworks/wikis/download/attachments/130515354/TpaeEcosystemDataIntegrationBestPractices.pdf?version=2

Important: As you review the *Data Integration Best Practices* white paper, consider following points:

- Though the white paper was authored soon after Tivoli's process automation engine 7.1 release, the description of the data loading tools, the design objectives for each of those tools, and their usage are applicable to Tivoli's process automation engine 7.5 release.
- In the white paper, a tool that is called the Content Catalog is referenced. With the SmartCloud Control Desk 7.5 release, this tool was renamed as the ISM Content Installer and can be used to import off-cycle content that is provided by IBM into a SmartCloud Control Desk 7.5 product environment.

1.2.2 Sequence of data loads

Most production environments are brought up the first time by using a combination of the data loading tools. Figure 1-1 shows a typical initial data loading effort that includes migration with the Migration Manager.

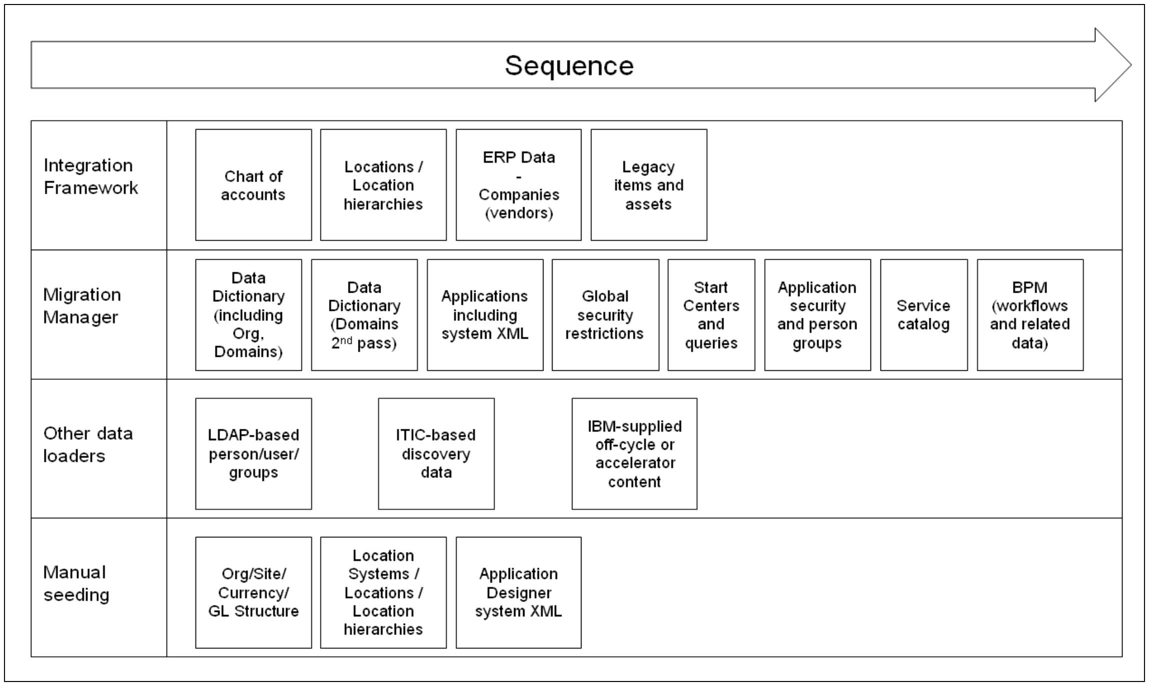

Sequence
Integration Framework
Chart of accounts
Locations / Location hierarchies
ERP Data - Companies (vendors)
Legacy items and assets
Migration Manager
Data Dictionary (including Org, Domains)
Data Dictionary (Domains 2nd pass)
Applications including system XML
Global security restrictions
Start Centers and queries
Application security and person groups
Service catalog
BPM (workflows and related data)
Other data loaders
LDAP-based person/user/ groups
ITIC-based discovery data
IBM-supplied off-cycle or accelerator content
Manual seeding
Org/Site/ Currency/ GL Structure
Location Systems / Locations / Location hierarchies
Application Designer system XML

Figure 1-1 Typical sequence of data loads and migration

The following steps show a more detailed description of the typical load or migration sequence:

1. Manually seed the organization names and currency.
2. Manually set up the GL account structure (segments).
3. Manually create the default chart of accounts for each defined organization.
4. Manually define the location system.
5. By using Integration Framework object structures, load the physical location hierarchy, including regions.
6. Run Integrity Checker, and look for any errors.
7. Correct the errors manually or by using Integrity Checker in repair mode.
8. After the manual steps are complete, the enterprise LDAP directory server is synchronized into the product, if the client uses directories. Person, user, and security groups can be loaded.
9. If Lightweight Directory Access Protocol (LDAP) is not part of your environment, migrate person, person groups, and security groups configurations, including employee and supervisory (management reports) information.
10. Migrate the data dictionary, including the domains.
11. Migrate the crossover and table domains to complete the data dictionary migration.
12. If the production environment includes SmartCloud Control Desk, run the service catalog object structure migration. The structural migration is complete.
13. Load the financial chart of accounts.
14. Load the currencies and exchange rates.
15. Migrate all of the application configurations, including menus and lookups.
16. Migrate application security configurations, including signature options.
17. Migrate the Start Centers and queries.
18. Migrate the SmartCloud Control Desk service catalog templates.
19. Migrate the SmartCloud Control Desk service catalog single service configuration.
20. Migrate the workflows and their associated roles, actions, and communication templates.
21. Run Integrity Checker, and look for and correct, any errors.

This sequence captures the essence of the data loading and migration steps. A number of tools are used when the steps in the sequence are performed. Experience with these tools accelerates the preparation of the production environment. Documenting this sequence during the planning stage is vital to the success of the overall migration effort. Ad hoc tasks that are performed without prior knowledge of the tools, applications, and content slow down the migration effort and lead to a reduction in customer satisfaction.

1.3 Content strategy

Migration is a consequence of configuration activity. Configuration activity creates content in the production database by using various configuration applications. Part of this content might be provided by IBM with the product, and other content might be created in the client's development environment. Visibility into and control of these configurations are important to ensure the consequent migration effort. Figure 1-2 shows the interaction of configurations and migrations. Content is the underlying artifact that is managed.

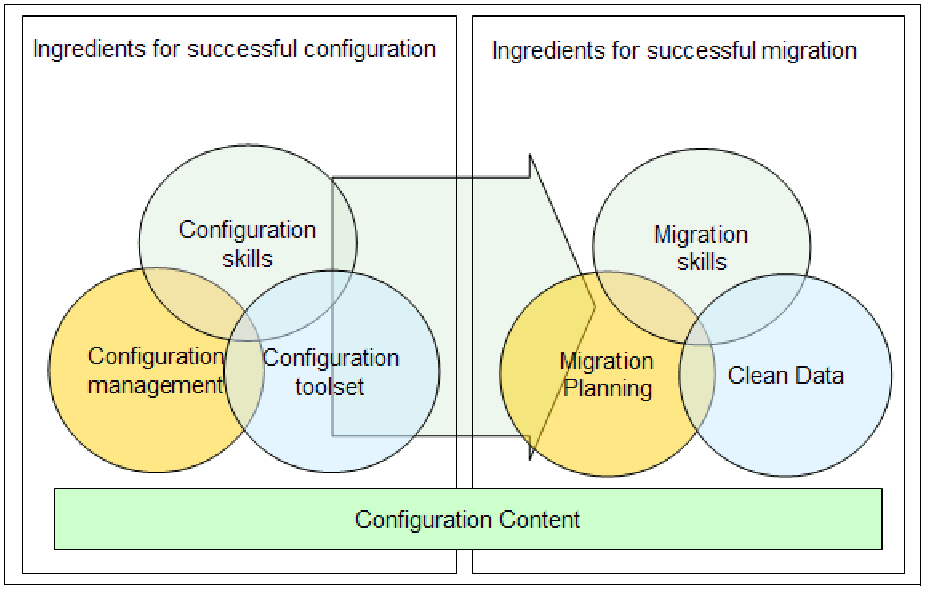


Figure 1-2 Interaction of configuration and migration

From the configuration perspective, configuration skills, configuration management, and the proper use of the configuration tool set are key to complete an efficient configuration in the client's development environment. Better configuration management practices lead to better migration planning.

Consistent use of the configuration applications and tools results in cleaner data that, in turn, accelerates the migration effort.

1.3.1 Dependencies among configuration content

Few product configurations are self-contained. Most product configurations have dependencies on other configurations. Product validations enforce these dependencies when configuration content is loaded into a production database. The dependencies among configurations and the validations that enforce them affect the migration effort. Figure 1-3 shows various dependencies among the configurations that drive a significant amount of product functionality.

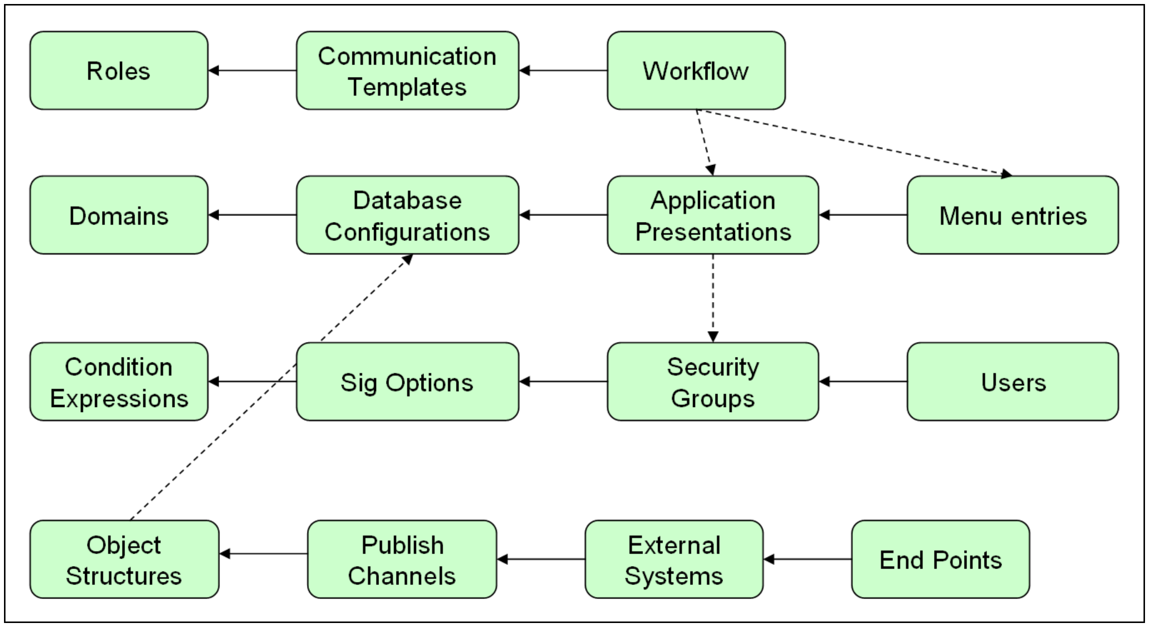


Figure 1-3 Dependencies among configurations

As shown in Figure 1-3, a workflow process configuration depends on communication templates and roles. Application presentations rely on business objects whose attributes, in turn, are associated with value lists in the form of domains. Security groups are associated with signature options, which, in turn, might be conditionally applied to the security group that is based on conditional expressions. External systems that are identified by using the Integration Framework are associated with Publish Channels, which, in turn, are supported by object structures.

Beyond the dependencies among logically related configurations, dependencies might exist between two different sets of configurations. In Figure 1-3 on page 9, a workflow process might depend upon a particular application presentation dialog or a menu entry that is presented during user interaction with the workflow process. Integration object structures are dependent upon the current business object definition.

Important: Because of the deep dependencies among configurations, sequencing the migration is critical.

1.3.2 Best practice content

It is a Tivoli direction to provide best practice content with the product or, periodically, through the Integrated Service Management (ISM) Library to help you get the product up and running quickly and you can start using the product before you engage in specific configurations and customizations. The content was designed based on customer implementation experiences. One example is the Optional Content that is offered with the SmartCloud Control Desk product.

The installation of Optional Content can include sets of starter content for Service Desk and Service Catalog implementation and content for configuration, change, and release management business processes that cover the typical IT Infrastructure Library (ITIL) situations. Service Desk optional content can include Security Groups, Start Centers, KPIs, and Queries. The PMUSRMGR security group, which is defined as part of service desk content, is intended to enable the role of User Contact Manager security group and is associated with the PMUSRMGR Start Center. Members of the User Contact Manager security group are responsible for managing the user contact in the context of processing service requests.

For more information about SmartCloud Control Desk optional content, see this website:

http://pic.dhe.ibm.com/infocenter/tivihelp/v50r1/topic/com.ibm.tusc.doc/content/content_intro.html

Another example of content is the IBM-provided Process Content Packs. For example, the Service Catalog process content pack offers production-ready set of Service catalogs, offerings, and supporting security groups. The use of this content pack enables clients to incorporate catalogs and offerings rapidly in their SmartCloud Control Desk product environment rather than designing from scratch.

For more information about content packs, see the Integrated Service Management library at this website:

`https://www-304.ibm.com/software/brandcatalog/ismlibrary/`

Recent clients loaded this content only to opt for more configuration and customization, which raises the issue of deciding what to do with this content when it comes to migration. Obviously, you have a few choices: create migration packages that exclude the predefined offerings, use currently supplied tools to add these offerings to the client's organization, or delete them.

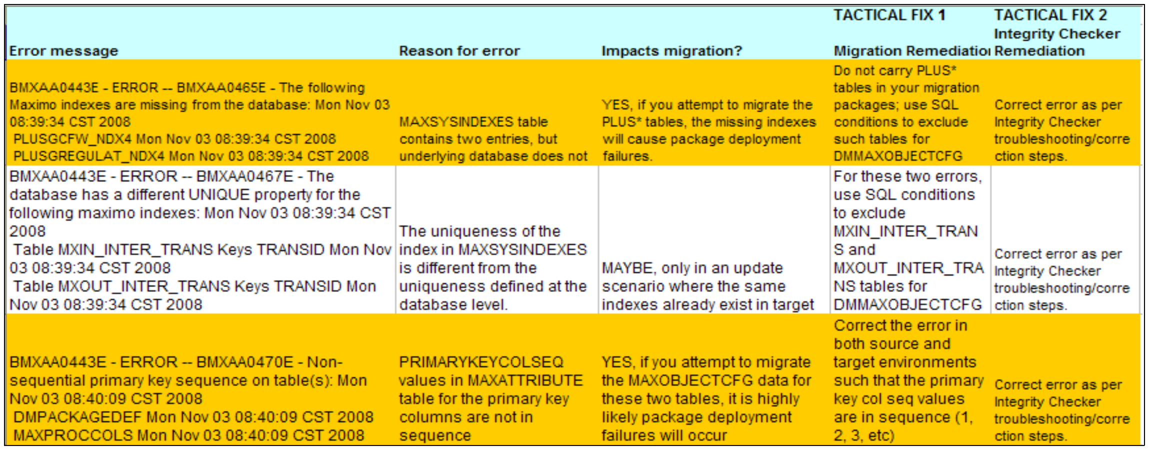
Another example is the use of Intellectual Capital (iCAP) to provide best practice business process content (workflow-based) that adheres to ITIL guidelines. Because this iCAP is loaded at installation on all environments, it is only the subsequent client modifications that must be promoted through the environment.

1.3.3 Integrity of configuration content

This section describes the need to establish the integrity of configurations in the production database by using the IBM-provided tool.

The Integrity Checker is the primary tool that is used to determine the integrity of many configurations. Although the Integrity Checker can be used during product upgrades, it is advantageous to also use it to assess integrity in a product environment, even after successful upgrade from an older release. The Integrity Checker tool is shipped with the product and can be started from the console command line. It produces a detailed log that includes warnings and errors that pertain to data integrity. It is important to continuously monitor the integrity of the underlying production database by periodically running the Integrity Checker in all of the production environments.

A best practice is to capture the Integrity Checker reported errors in a spreadsheet form, review the error messages, and identify the resolution of those errors. By preparing this spreadsheet, you can assess the impact that the error has on migration. Figure 1-4 shows a sample spreadsheet of collected error messages and includes resolutions.



			TACTICAL FIX 1	TACTICAL FIX 2
Error message	Reason for error	Impacts migration?	Migration Remediatioı	Integrity Checker Remediation
BMXAA0443E - ERROR -- BMXAA0465E - The following Maximo indexes are missing from the database: Mon Nov 03 08:39:34 CST 2008 PLUSGCFW_NDX4 Mon Nov 03 08:39:34 CST 2008 PLUSGREGULAT_NDX4 Mon Nov 03 08:39:34 CST 2008	MAXSYSINDEXES table contains two entries, but underlying database does not	YES, if you attempt to migrate the PLUS* tables, the missing indexes will cause package deployment failures.	Do not carry PLUS* tables in your migration packages; use SQL conditions to exclude such tables for DMMAXOBJECTCFG	Correct error as per Integrity Checker troubleshooting/correction steps.
BMXAA0443E - ERROR -- BMXAA0467E - The database has a different UNIQUE property for the following maximo indexes: Mon Nov 03 08:39:34 CST 2008 Table MXIN_INTER_TRANS Keys TRANSID Mon Nov 03 08:39:34 CST 2008 Table MXOUT_INTER_TRANS Keys TRANSID Mon Nov 03 08:39:34 CST 2008	The uniqueness of the index in MAXSYSINDEXES is different from the uniqueness defined at the database level.	MAYBE, only in an update scenario where the same indexes already exist in target	For these two errors, use SQL conditions to exclude MXIN_INTER_TRANS and MXOUT_INTER_TRANS tables for DMMAXOBJECTCFG	Correct error as per Integrity Checker troubleshooting/correction steps.
BMXAA0443E - ERROR -- BMXAA0470E - Non-sequential primary key sequence on table(s): Mon Nov 03 08:40:09 CST 2008 DMPACKAGEDEF Mon Nov 03 08:40:09 CST 2008 MAXPROCCOLS Mon Nov 03 08:40:09 CST 2008	PRIMARYKEYCOLSEQ values in MAXATTRIBUTE table for the primary key columns are not in sequence	YES, if you attempt to migrate the MAXOBJECTCFG data for these two tables, it is highly likely package deployment failures will occur	Correct the error in both source and target environments such that the primary key col seq values are in sequence (1, 2, 3, etc)	Correct error as per Integrity Checker troubleshooting/correction steps.

Figure 1-4 Assessment of Integrity Checker errors

As shown in n Figure 1-4, the spreadsheet includes the following columns:

- Error Message: Reproduces the Integrity Checker error as retrieved from the Integrity Checker log file.
- The Reason for Error: Reproduces the Integrity Checker product documentation by correlating the error message number with the Integrity Checker error tables.
- The Impacts Migration: Makes a determination if the particular error affects the Migration Manager activities.
- The TACTICAL FIX 1: Determines whether the Migration Manager can be controlled to avoid migration failures because of the error.
- The TACTICAL FIX 2: Determines whether the resolution is to fix the integrity error directly in the underlying database and then migrate.

In most cases, it is necessary to fix the integrity error in the database and then migrate.

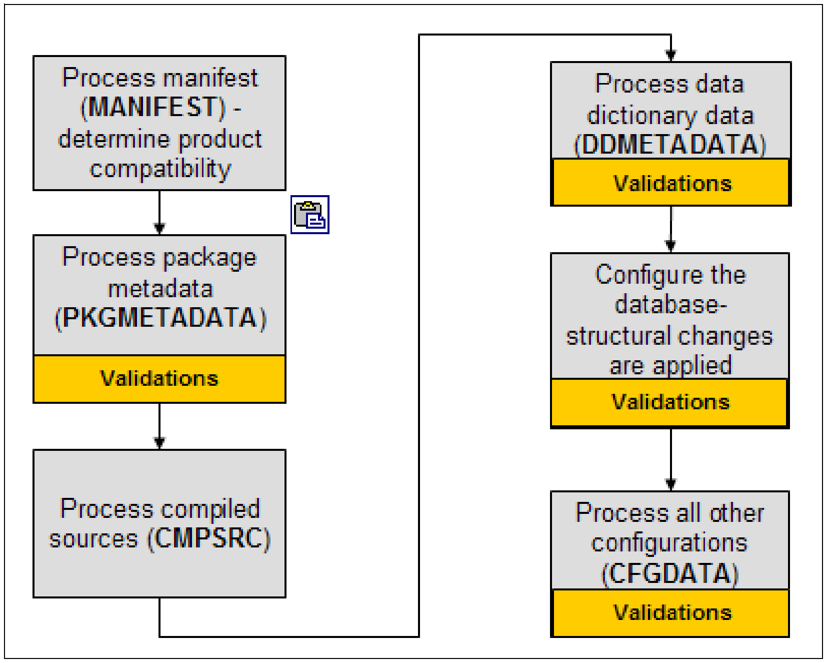
To understand and use the Integrity Checker tool, see the detailed product documentation at this website:

http://www-01.ibm.com/support/docview.wss?uid=swg21266217&aid=3

1.3.4 Content validation

This section describes the importance and impact of validation to the migration effort.

The configuration contents of a migration package are deployed into a production environment by using the Migration Manager. The deployment consists of a well-defined automated procedure of extracting the contents of the migration package and processing those contents in a specific order. Depending on the type of content, the procedure is temporarily stopped to allow an administrator to perform manual tasks and the procedure is resumed when those manual tasks are complete. All content that is deployed into the underlying production database is subject to validation. Such validation ensures that the integrity of the production database is always preserved. This validation cannot be turned off. Figure 1-5 shows the role of validation in deployment.



Process manifest (MANIFEST) - determine product compatibility
Process package metadata (PKGMETADATA)
Validations
Process compiled sources (CMPSRC)
Process data dictionary data (DDMETADATA)
Validations
Configure the database-structural changes are applied
Validations
Process all other configurations (CFGDATA)
Validations

Figure 1-5 Role of validation in package deployment

As shown in Figure 1-5, key validations are applied to the data dictionary configurations, the database configuration process, and all other configurations, such as person records, security group records, and workflow processes.

When validations fail, the Migration Manager reports a deployment error. A deployment error represents the inability of the configuration application to successfully apply the data validation rules to the particular configuration that is deployed. It is the responsibility of the Migration Manager to report deployment errors that result from validation failures.

The root cause of these validation errors can be one or a combination of the following reasons:

- Incomplete packaging of the configuration content
- Incorrect sequencing of migration
- Bad data in the Source production environment
- Limitation of the Migration Manager
- Defect in the product

In each case, the appropriate remedial measures are taken to ensure recovery from the error and continuation of the migration effort.

Incorrect sequencing or incomplete packaging can be addressed by improving product skills. Bad data often is the result of loading data directly into the database by using database commands. This practice must be avoided at all costs. Bad data also can be the result of database scripts that are supplied with the product. Part of the bad data can be corrected by using Integrity Checker. If bad data was shipped with the product, IBM treats this situation as a defect. The Migration Manager limitations can be addressed by adopting other tools to load the configuration or the related data that is needed. Product defects are addressed by IBM by using standard procedures. Understanding the nature of the deployment error helps to resolve the error.

1.3.5 Content and source control systems

The role of source control systems is a topic that is often raised by practitioners and clients when configurations are managed and migrations are performed. Tivoli’s process automation engine does not offer any built-in ability to create a version. Configurations are saved to the designated tables in the underlying production database, which overwrites any previous values. How can configuration versions be maintained? Does the Migration Manager play a role?

The Migration Manager does not enable or mandate a particular source control model. However, from a change management perspective, implementing a source control model or extending an existing source control model can yield benefits to practitioners and clients that extend beyond the migration effort.

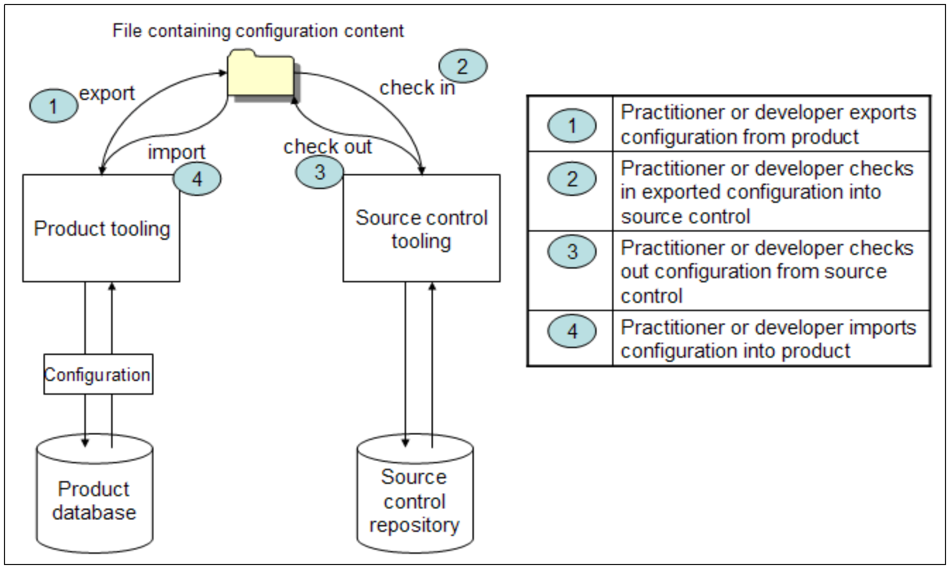
In this section, we describe the following source control models:

- Manage versions of configurations with source control
- Publish migration packages through source control

Managing versions of configurations with source control

Part of the configuration content in the product is shared among team members. Multiple edits might be performed in a short time on this content. When the configuration content is stored in the production database, the changes that are made by the last team member who saved the configuration are alone written into the database. This process can be a significant change management challenge in shared product development environments. If there is a need to revert to a previous revision of the configuration content, the only way to revert to a previous version is through the implementation of a source control model.

In this source control model, chosen configuration content is exported as a file from the production environment and checked into a source control system. Practitioners or developers that require a change to that content must check out that file, import it into the production environment, and implement the change by using the appropriate application. After it is complete, the configuration is exported again and placed back into source control. This model is shown in Figure 1-6.



1	Practitioner or developer exports configuration from product
2	Practitioner or developer checks in exported configuration into source control
3	Practitioner or developer checks out configuration from source control
4	Practitioner or developer imports configuration into product

Figure 1-6 Source control model to manage versions

Although cumbersome, implementing this model ensures that changes are traceable and that the authors of those changes are accountable during the implementation cycle. This approach of managing revisions of a configuration must be considered as part of the overall content strategy.

This model is commonly applied to major system configurations such as objects, domains, start centers, and presentations. It is also supplied to shared Application Designer configurations, such as library, lookups, and menus. Other system configurations might be chosen to participate in version control, depending upon the needs of clients.

With the release of Tivoli's process automation engine V7.5, the Migration Collections feature is a first step in aligning system configurations with a version control model. A migration collection that represents a particular unit of functionality or a defect fix contains a specified set of system configurations. An XML-representation of each system configuration can be retrieved by using Migration Collections and saved as a file into the implementer’s workstation. For more information about the Migration Collection’s export and import feature and how it could be used to facilitate version control, see Chapter 3, “Migration collections” on page 49.

Publishing migration packages through source control

Certain clients also choose to check in to source control systems entire migration packages in the form of compressed files that are provided by the Migration Manager. The rationale behind this approach is that the source control system alone holds the authorized set of migration packages that must be used to migrate. Direct distribution of packages from a source environment to a target environment via file system or database distribution is not allowed. This model is shown in Figure 1-7.

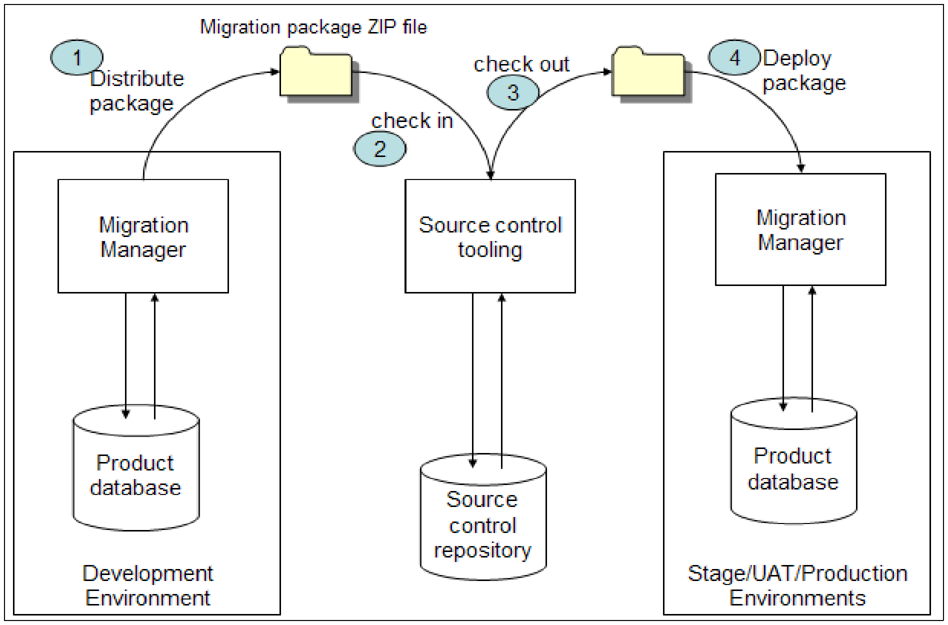


Figure 1-7 Source control model to publish migration packages

We recommend flexibility regarding this practice. The overall implementation time line and resource constraints must be taken into account before this approach is implemented. The practice is more easily implemented with clients who already have an established process of distributing content through source control systems.

Of the two source control models that are presented here, the need to manage the versions of key configurations is more critical to the success of the development effort. An inability to successfully manage shared configurations can lead to a breakdown of the development effort.

1.4 Migration approach

In this section, we describe the recommended migration approach. The release of Tivoli's process automation engine 7.5, Migration Manager offers the following modes of operation:

- Snapshot
- Change
- Collection

1.4.1 Snapshot mode characteristics

The Snapshot mode is a view of the system at the point of creating a package. This mode is often used because the Migration Manager was used late in the program and therefore, the Snapshot sees all changes that were made since the installation. The disadvantage with Snapshot is that you migrate everything (without any filtering) or the implementer must define filters to select only the changes made. By using naming conventions, such as a client prefix on all structural and other database changes, it is possible to write these Where clauses and limit the migration package size to close to the size that is created by a corresponding Change mode package, for a set of changes.

Consider the following characteristics of Snapshot packages:

- The Snapshot package can filter (one-to-many) and migrate complete configuration content, which ensures Source and Target synchronization.
- When filters are defined, SQL criteria must be defined and the implementer must have the requisite SQL skills and in-depth knowledge of the data model to define the appropriate, valid filter.
- Snapshot processing actions can be specified that determine whether the configuration content is fully replaced or merged into the Target production environment.
- The entire migration requirement is clearly defined, identified, and documented, to the exact names of the content elements.
- The source environment contains a considerable number and amounts of configurations that took significant time to develop, a bulk migration is appropriate, and all of the previous characteristics are true or wanted.

1.4.2 Change mode characteristics

After it is enabled, the Change mode incorporates event listeners that support tracking changes that are made to business objects up until the time when it is disabled. Change mode tracks who does what and when. Content is extracted based only on the tracked changes.

Consider the following characteristics of Change packages:

- Useful when a source environment and a target environment exist and are synchronized at the time that this type of package is activated (for example, when the last migration is done before production, as described previously).
- Useful when active packages do not overlap (listening to the same objects), development efforts also do not overlap (each development effort uses its own Change packages), and all of the previous characteristics are true or wanted.
- When too many development efforts overlap and it is difficult to define non-overlapping packages, a single Change package is useful to track the changes and define the migration requirements. This use is a hybrid use of Change mode where the Change package is used merely as a tracking device. The actual migration is performed by using Snapshot packages.

For more information, see 15.6, “Change tracking and ad hoc reporting” on page 369.

1.4.3 Collection mode characteristics

In Tivoli's process automation engine 7.5 release, a new mode of packaging was introduced called Migration Collections. A migration collection is a collection of system configurations that should be migrated to a target product environment. Unlike Snapshot or Change modes, a migration collection is built up over time by implementers that use a pick and choose approach. During the time the migration collection is built, the implementer is not concerned about packaging. Instead, the focus is to ensure the complete collection of system configurations. When a migration collection is deemed complete, a collection package definition can be generated from the Migration Collections application.

Consider the following characteristics of Collection packages:

- Collection packages are not manually defined; they are generated from a particular migration collection. The set of configurations that is included in the migration collection drive the package definition.
- After a migration collection is defined, system configurations can be added to the collection by using multiple options. The particular option that is used at a point in time depends upon the implementer's needs to shadow the development effort.
- Unlike a Snapshot package, Collection packages do not require or support the use of filters. The contents of the particular collection fully determines the content of the Collection package.
- One of the migration collection options is to enable tracking changes to system configurations similar to a Change package.

For more information, see Chapter 3, "Migration collections" on page 49.

The migration scenarios in this document are implemented by using one or a combination of the three modes to illustrate the effective use of Migration Manager. Some scenarios migrate the same content by using a different mode each time to compare and contrast the modes.

1.4.4 Comparing the migration approaches

A number of factors influence the implementer's decision to use a particular package mode. Making the following right decisions results in accelerated migration, which benefits the business, its operations, and the users:

- Granularity: Does the delivery of application functionality and defect fixes require granular identification and collection of system configurations?
- Role separation: Is the development role completely or substantially independent of the packaging and migration role?
- Development cycle: Are business applications rolled out to the users in an incremental manner? Is the development cycle iterative?
- Implementer skills: Does the role that is invested with packaging and migration possess deep knowledge of the product data models, relationships, and SQL commands?
- Traceability requirement: From a project traceability perspective, is there a requirement to trace system configurations and changes to a particular owner and development iteration?
- Deletion of configurations: Will Migration Manager be used to trigger deletions of configurations in a target environment?

Table 1-2 provides a matrix of these factors and identifies the corresponding package modes that best fit the migration efforts.

Table 1-2 Package mode decision matrix for migration strategy

	Project characteristics	**Benefits**
Use Change if...	Changes in the development environment must be tracked at all levels of a system configuration and those exact changes, including deletes that are reproduced in a target environment.	Change offers a granular tracking capability, including delete events of all business objects that comprise a system configuration. This mode is the only mode that can propagate deletes of entire system configuration from a source to target environment.
Use Snapshot if...	Granularity of assembling system configurations is not a key requirement.	Effective use of SQL criteria ensures that system configurations can be extracted in bulk into a package. This ability would also reduce the number of packages that must be designed.
	Development and migration role are fulfilled substantially by the same resource.	Avoid two-step approach where a migration collection must be assembled before a migration package is created.
	Traceability into the set of system configurations that comprise a defect fix or functionality is not a key requirement.	Avoid two-step approach where a migration collection must be assembled that represents the fix or function before a migration package is created.
Use Migration Collections if...	System configurations are identified and assembled in a granular manner; some clients call this approach “pick and choose”.	Migration collections can be designed to closely match the specific project deliverables. For example, the specific system configurations that comprise a fix for a defect.
	Development role is completely or substantially independent of the packaging and migration role.	Development roles can collect the system configurations into designated migration collections; hand-off to packaging roles.
	Implementers do not possess adequate skills to prepare complex SQL criteria or deep knowledge of the product data model, relationships, and individual business objects and attributes.	Implementers can, over time, collect individual system configurations by using various intuitive Migration collection constructs rather than preparing snapshots with SQL criteria that is defined in trial-and-error manner.

	Project characteristics	**Benefits**
	Traceability is a critical requirement.	By defining migration collections to match a particular project deliverable (functionality or defect), the specific system configurations that are added to a migration collection offer greater visibility into what was created or modified in the development environment.

1.4.5 Hybrid migration approaches

Based on the authors' collective experience in supporting customer implementations, there is not a one-size-fits-all approach in promoting system configurations to production (or any target environment). Migration Manager should be considered as a set of tools from which the appropriate tools are chosen to meet the requirements of the project implementation.

In a source environment where system configurations are assembled into a package, it is more than likely that a combination of the Migration Manager modes is used to maintain development and packaging velocity and ensure timeliness of package delivery to the designated target environment. The following potential hybrid approaches are highlighted:

- Migration collection and change approach
- Migration collection with comparison and change approach
- Snapshot and change approach

Migration collection and change approach

The *Migration collection and change approach* uses migration collections to address new functionality and defect fixes that are prepared in a source environment. The "pick and choose" approach ensures that the required system configurations are placed in the appropriate migration collection that represents functionality or fix. Change package is used to capture deletions of system configurations and propagate them to the target environment. With this approach, each collection results in a collection package. The approach is adopted for pre-production and post-production maintenance promotions.

Figure 1-8 shows the migration collection and change approach.

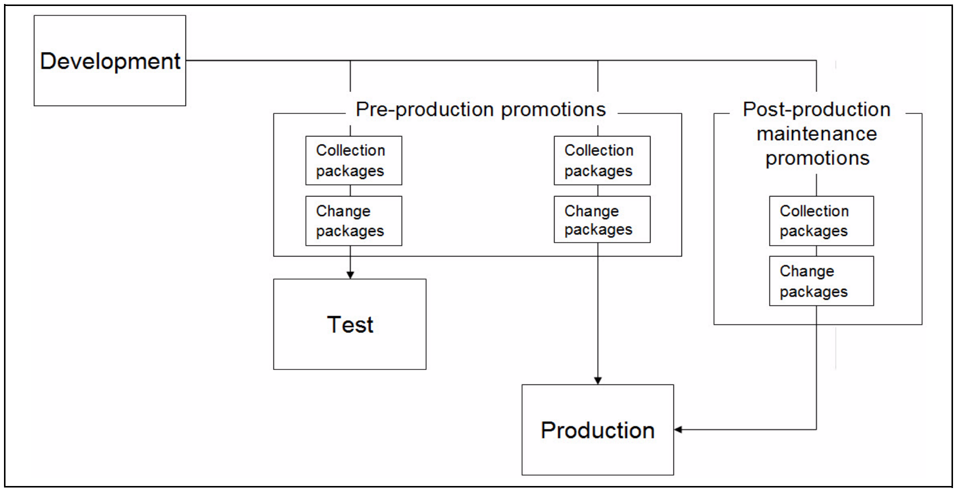


Figure 1-8 Migration collection and change approach

Migration collection with comparison and change approach

The *Migration collection with comparison and change approach* uses migration collections to address new functionality and defect fixes. Change package is used to capture deletions of system configurations. However, for post-production promotions, the source and target are compared and comparison results are placed into an appropriate migration collection for promotion. Comparisons also could be used earlier in the delivery cycle to promote system configurations to test environments. This approach effectively reduces the burden on the implementer to prepare collections from scratch. It is the quickest way to prepare a migration collection when source and target environments are available.

Figure 1-9 shows the collection with comparison approach. The dotted lines from the comparison results boxes indicate the execution of Migration Manager's comparison functionality between the source and target environments. The line that connects comparison results with collection packages indicates the preparation of collections from comparison results.

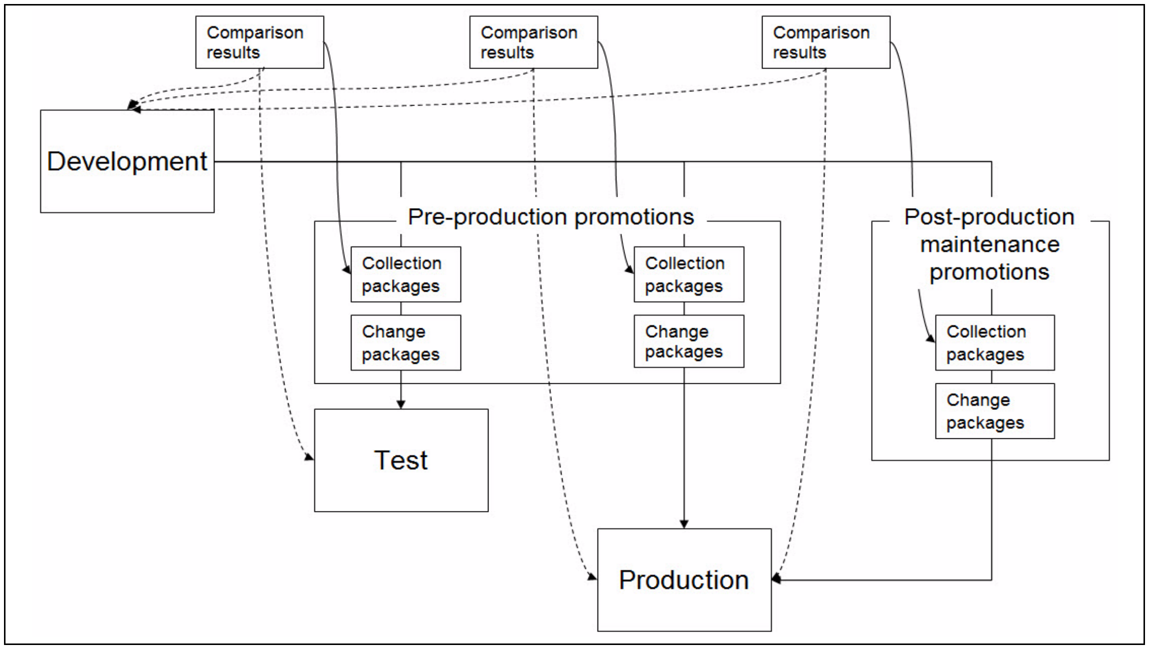


Figure 1-9 Collection with comparison approach

Snapshot and change approach

The *Snapshot and change approach* employs Change and Snapshot packages at the same time. The purpose is to capture change information automatically, which reduces the need for practitioners and developers to laboriously enter the individual configurations that they create into a document. Implementing this hybrid migration strategy carries a minimal cost (two package definitions instead of one) but yields significant benefit during the development and migration effort.

With this hybrid migration strategy, both package definitions identify the same types of content. The Change package is defined exclusively to track the same configuration content that eventually is packaged with the Snapshot package. An ad hoc report is configured to display the changes that are tracked by the particular Change package.

Figure 1-10 shows snapshot and change approach.

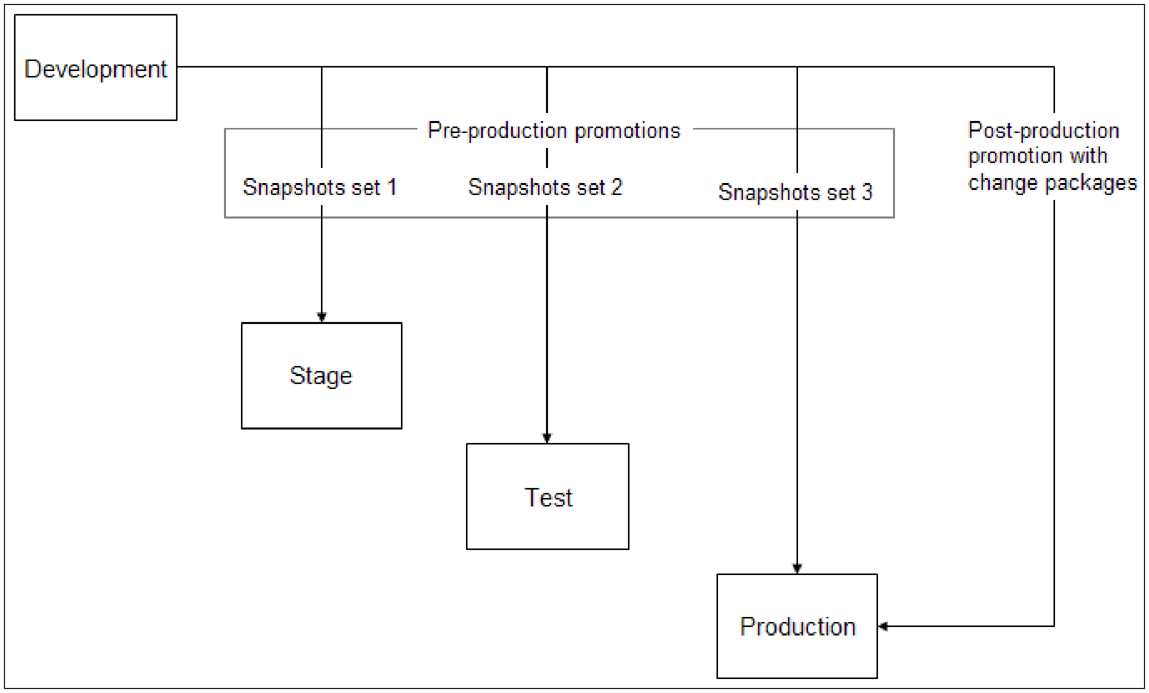


Figure 1-10 Migration modes

For more information, see 15.6, "Change tracking and ad hoc reporting" on page 369.

1.5 Migration planning

A migration plan is the formulation of the set of actions that promote configuration content from the development environment to upper environments, such as stage, test, production, or training. The migration plan manifests in the form of a controlled living document that serves as input to the Migration Manager tasks to be performed by a skilled resource. However, it is not limited to the Migration Manager tasks alone.

The plan also enumerates the following manual activities that are performed outside of the Migration Manager functionality:

- Creation of configurations by using the corresponding Configuration application; for example, organization.
- Execution of SQL scripts; for example, loading a set of person records.
- Execution of Integration Framework-based data loads; for example, loading a set of GL accounts or loading a set of company records from an existing system.
- Execution of build process for Java classes; for example, building the product's deployable EAR file.

The critical part of the plan is the sequence in which all of these tasks are performed and the unambiguous identification of task owners. The goal is to eliminate as many unknowns as possible and streamline the preparation and delivery of the upper environments to their users.

The plan facilitates the migration before the production rollout, and periodic migrations after the production rollout is complete.

Creating and maintaining a migration plan document offers the following benefits:

- The scope of the migration is clearly established.
- All required tools and skills are clearly identified.
- Owners of all tasks are clearly identified.
- The progress of the migration is monitored.
- Status updates are provided to stakeholders.

This IBM Redbooks publication provides guidance to the content of a migration plan document. You also can use the samples that are provided with this book. For more information about downloading these samples, see Appendix A, "Additional material" on page 445.

1.5.1 Contents of the migration plan

The migration plan document benefits from a spreadsheet format. The spreadsheet is created and placed in a shared environment. Certain practitioners even place the spreadsheet under revision control so that separate versions of the spreadsheet can be maintained and retrieved. Placing the spreadsheet under revision control enables stakeholders to compare multiple versions of the spreadsheet and ensure authorized access and updates to the controlled document.

There might be many variations of the spreadsheet that are based on the practitioners' approach, requirements of clients, and policies of IT-based organizations.

The migration plan typically includes the following areas:

- Tasks
- Task sequence
- Owners
- Tools
- Dates

The following sections describe a reasonable and reusable spreadsheet-based migration plan.

The spreadsheet contains the following worksheets:

- Migration plan revision history
- Migration environments, products, and versions
- Migration tasks

Migration plan revision history

This worksheet records changes to the migration plan and the authors of each of those changes. Table 1-3 shows the columns that record revision history.

Table 1-3 Record revision history

Revision	Date	Author	Description	Reviewed by	Approved by

This worksheet provides the following benefits:

- Provides an audit trail of changes to the plan
- Ensures accountability through review and approval information

Migration environments, products, and versions

This worksheet records each migration environment that is part of the IT landscape. At a minimum, this worksheet includes the development, test, and production environments. For each environment, this worksheet shows the list of installed products. This information is easily obtained from the production environment's web user interface (System Information menu item in the user's Start Center). In addition, the type of the environment is recorded; that is, a particular environment is a Source or Target from the migration perspective. Table 1-4 provides example information for the environment, products, and revisions.

Table 1-4 Columns that record the environment, products, and revision information

Environment type	**Products and versions**	**Product URL**	**Migration package distribution**	**Package file location**
Development	SRM Problem Management 7.2.0.1 Build 20100316D2 DB Build V7201-01 SRM Incident Management 7.2.0.1 Build 20100316D2 DB Build V7201-01 SRM Service Request Management 7.2.0.1 Build 20100316D2 DB Build V7201-01 IBM Tivoli Common Process Components 7.2.0.01 Build 20100109D DB Build V7201-02 Base Services 7.1.1.6-LA20100312-0937 Build 20091208-1415 DB Build V7116-173 HFDB Build HF7116-04	http://hostname:9999/maximo	FILE	H:\share\migration\packages

This worksheet provides the following benefits:

- Provides at-a-glance information about the IT environment to stakeholders
- Avoids confusion among implementers about Sources and Targets
- Saves time when information about deployment issues is gathered
- Avoids process breakdown when implementation resources are transitioned

Migration tasks

This worksheet is the most critical element of the plan. This worksheet records the following facets that pertain to the migration effort:

- Manual data entry into the production environment
- Import of existing data into the production environment by using Integration Framework
- Deployment of migration packages into the production environment by using the Migration Manager
- Necessary application server restarts
- Special situations where the standard order of migration must be changed to accommodate dependent data

Table 1-5 shows sample content for this worksheet that is based on the recommended approach. The line items in the worksheet are representative only. Depending on the extent of configuration, the number of manual steps, the need to import data by using Integration Framework, and the number of migration packages that are required to complete the migration, the effort can vary. The worksheet is beneficial in reducing the uncertainty that is inherent in data migrations and accelerating the migration activities.

Table 1-5 Migration tasks

Sequence	Summary	Description	Method	Owner
1	Integrity Checker	Run Integrity Checker on Target database in report mode	Manual	Product implementer
2	Correct Integrity Checker errors and warnings	Use Integrity Checker with repair mode or manually correct with SQL tools	Manual	Product implementer
3	Database backup	Back up the Target production database	Manual	DBA
4	Prepare EAR	Build the product EAR file	Manual	Product implementer
5	Deploy EAR	Deploy the product EAR and restart the application server	Manual	Product implementer

6	Currencies and exchange rates	Load currencies and exchange rates	Manual	Product implementers
7	Org/currency	Seed the ORG names and currency	Manual	Product implementer
8	GL structure	Set up the GL account structure	Manual	Product implementer
9	Chart of accounts	Create a chart of accounts for each organization	Manual	Product implementer
10	Default chart of accounts	Create a default chart of accounts for each organization	Manual	Product implementer
11	Chart of accounts	Load more accounts via Integration Framework	Integration Framework	Migration Manager
12	System of locations	Define and load systems and locations	Manual or Integration Framework	Product implementer
13	Run Integrity Checker	Run Integrity Checker against the development database in report mode	Manual	Product implementer
14	Correct Integrity Checker errors and warnings	Use Integrity Checker with repair mode or manually correct with SQL tools	Manual	Product implementer, developers
15	Data dictionary (Phase 1)	Migrate conditional expressions	Package	Migration Manager
16	Data dictionary (Phase 2)	Migrate all domains except the table and crossover	Package	Migration Manager
17	Data dictionary (Phase 3)	Migrate data dictionary configurations, including objects and attributes	Package	Migration Manager
18	Data dictionary (Phase 4)	Migrate table and crossover domains	Package	Migration Manager
19	Automation Scripts (Phase 1)	Migrate automation scripts that support conditional expressions, business objects, and attribute validations	Package	Migration Manager
20	Applications	Migrate Application Designer system XMLs	Package	Application Designer

21	Applications	Migrate all applications and menus except system XMLs; server restart that is needed if production environment is clustered	Package	Migration Manager
22	Persons, users, and security groups	If Lightweight Directory Access Protocol (LDAP) is synchronized for an LDAP-enabled client, see Chapter 6, “Migrating security configuration data” on page 121	SYSTEM	Product implementer
23	Persons, users, and security groups	If LDAP is not enabled, migrate persons, users, and security groups	Package	Migration Manager
24	Global data restrictions	See Chapter 6, “Migrating security configuration data” on page 121	Package	Migration Manager
25	Application Security package	Application Security package for signature options and queries	Package	Migration Manager
26	Start Center	Start Center and Queries package; each individual user must update their Start Center to apply changes	Package	Migration Manager
27	Reporting (Phase 1)	Migrate Ad Hoc report object structures	Package	Migration Manager
28	Reporting (Phase 2)	Migrate BIRT report designs, reports, and KPIs	Package	Migration Manager
29	Cron tasks, system properties	Migrate cron task definitions, instances, and user-defined system properties	Package	Migration Manager
30	Integrations (Phase 1)	Migrate integration object structures	Package	Migration Manager
31	Integrations (Phase 2)	Migrate all other integration configurations	Package	Migration Manager
32	Service Catalog	Service Catalog Object Structure package	Package	Migration Manager
33	Business process management (BPM) (Phase 1)	BPM (workflow) package to migrate the workflows and their associated role, actions, and communication templates, except application actions that launch workflows	Package	Migration Manager

34	Automation Scripts (Phase 2)	Migrate automation scripts that support workflows, escalations, and UI-driven actions	Package	Migration Manager
35	BPM (Phase 2)	Migrate application actions that launch workflows	Package	Migration Manager
36	Post-migration task	Activate communication templates and workflows	Manual	Product implementers
37	Classifications	Migrate classifications	Package	Migration Manager
38	Service Catalog Templates	Service Catalog Templates package	Package	Migration Manager

After a plan document is created, it is recommend that the stakeholders meet to review the plan. This meeting disseminates the critical information to all participants and identifies missing or incorrect information in the plan document. Subsequent periodic meetings provide the continuous monitoring of the migration process.

Best practice: Adopt a well-defined naming convention for all system configurations that are created and managed in any development environment. Irrespective of the migration approach, the ability to quickly and clearly identify a particular system configuration vastly improves traceability in the product environments, enables more efficient communication, and promotes understanding of the implementation among project team members.

1.5.2 Working with migration time frames

The migration effort is always measured by clients in terms of the time that is used, which must be adequately scoped out as part of the project plan.

In the pre-production phase of a project, the development of configurations can take several weeks and involve multiple practitioners and developers. It is important for the migration planning and activities to remain in lockstep with the development activities during this phase. The most common mistake that is made is to defer the migration activities until after the development is complete. Most clients plan for a few days during which time they expect migration activities to complete and the production environment to be ready for user access. Client expectations of migration cannot be met when development and migration activities are separated.

Best practice: In the pre-production phase of a project, the migration planning document must be complete at the same time that the development and configuration activities for the project complete.

In the post-production phase, certain configuration and development activity is ongoing. This activity is the result of feedback from users or application managers. A smaller scale migration is undertaken to promote this maintenance or incremental configurations to the existing production environment. Most clients expect this migration to begin and complete within specific maintenance windows. The most commonly encountered maintenance windows are two, four, or six hours. The maintenance windows are typically opened at the onset of a weekend when user activity is minimal and can be stopped without affecting the business. It is critical that the maintenance activities complete within the maintenance window that is specified by the clients. Developing a maintenance window migration plan is useful. This plan can be a simpler version of the pre-production plan that is outlined in earlier sections.

A number of factors affect the duration of all types of migrations that might result in the implementation exceeding the expected migration time frames. The following factors use a considerable amount of time, not directly involving the Migration Manager or Integration Framework:

- Application server restarts

  Production environments typically use multiple Java virtual machines (JVMs), which run in clustered application servers. There might be more JVMs outside of the cluster that supports cron tasks, integration, and reporting functions. Shutting down and restarting all of the JVMs can use a significant amount of time that reduces the migration time frame.

- Post-migration tasks

  After a migration completes, manual tasks are typically performed, such as the activation of workflow processes, escalations, and cron tasks. These tasks are done through the product’s application user interface (UI) and use a certain portion of the migration time frame.

- Running tools

  Some tools, including the Integrity Checker, might be run periodically. Depending on the size of the production database, the tool can take a longer or shorter time to complete, yet use a certain portion of the migration time frame.

- Other factors

  Database administrators might be required to review structural changes to the underlying production database and, if required, to modify database objects, such as indexes and table spaces. The intent is to ensure the optimal performance of the database. These activities use a certain portion of the migration time frame.

Figure 1-11 shows a breakdown of the various types of tasks as a percentage of the total migration effort. This chart helps to establish the fact that the Migration Manager is not the only tool that is used to achieve total migration.

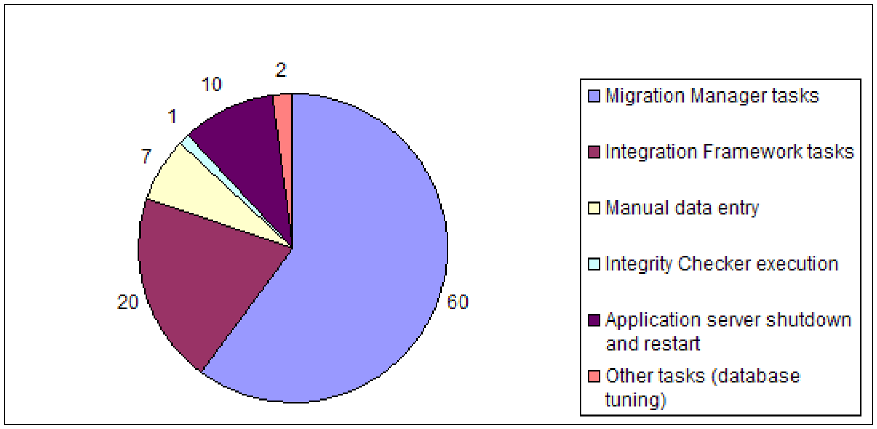


Figure 1-11 Migration effort breakdown

Keeping in mind these factors, managing the migration effort is as much managing the client's expectations with multiple constraints as performing the actual migration tasks by using the Migration Manager. Up-front discussion and agreement with the clients can go a long way in ensuring a successful and timely migration.

1.6 Migration Manager reference material

You can find the list of reference material that is available for the Migration Manager in Appendix A, “Additional material” on page 445. These links are mostly the Technotes or webdocs that are published on IBM Tivoli Support websites about the Migration Manager.

Important: Although many of the documents in this list were published against an older release of Tivoli's process automation engine, they remain valid for Tivoli's process automation engine 7.5 release and all products that were built with this release.

2

What is new in Migration Manager

This chapter describes the features and improvements that were added to Migration Manager in the 7.5 version of Tivoli's process automation engine. These features and improvements were identified in response to implementer and administrator usage and feedback. They are designed to meet the following requirements:

- Accelerate migration by reducing the burden on developers and implementers to organize and design packages
- Better support traceability of system configurations in development environments
- Improve the usability of the migration application user interfaces

By delivering features and improving usability against these requirements, the Migration Manager is now a powerful toolset that implementers and administrators can adapt to varying migration needs. The benefit to organizations that are implementing products that are built on Tivoli's process automation engine is accelerated value to their users through quicker production roll outs and lower Information Technology (IT) costs.

This chapter includes the following sections:

- Migration Manager functions
- Accelerated package design and creation features
- Deployment management features
- Usability improvements
- Migration Manager new and enhanced functions summary

2.1 Migration Manager functions

In the previous releases of Tivoli's process automation engine, Migration Manager delivered the following broad sets of functions:

- Package management functions
- Content management functions

Package management functions enable the definition, creation, distribution, and deployment of packages that contain system configuration data. Content management functions, in the form of object structures and migration groups, support the extraction of system configuration data from a source environment into a package. Object structures determine the XML-representation of a system configuration that is extracted into a package. Migration groups determine the logical groupings of system configurations that often are packaged together. Figure 2-1 shows these two sets of functions and how they tie into each other.

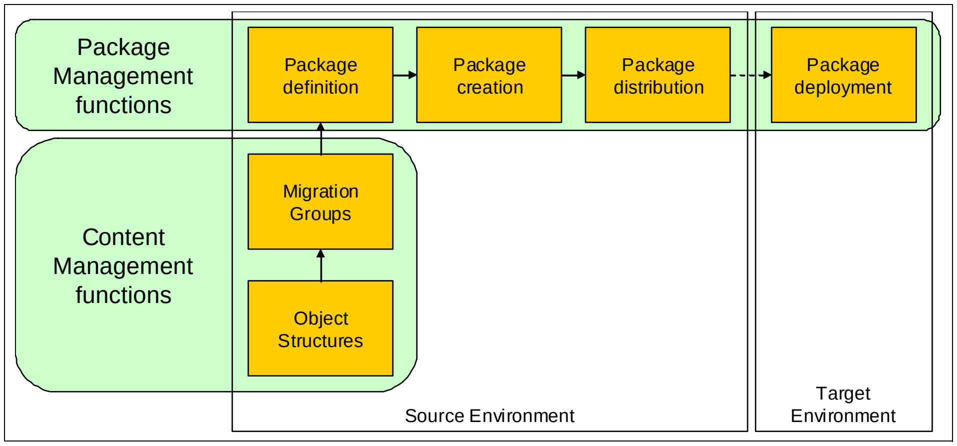


Figure 2-1 Package and content management functions

In Tivoli's process automation engine 7.5 release, these functions are enhanced and new functions were added to accelerate migration. The following sections describe the new enhancements and functions.

2.2 Accelerated package design and creation features

Migration Manager packages are defined by administrators and implementers to represent a well-defined set of system configurations that must be promoted from one product environment to another. The design of packages requires the implementer to be fully aware of the development activities and to identify and group system configurations that result from those development activities.

In the previous release of Tivoli's process automation engine, Migration Manager offered snapshot and change package features to identify and collect system configurations. Considerable skill and planning were required to design the appropriate package definition to capture the wanted system configurations.

With Tivoli's process automation engine 7.5 release, focus shifted to the approach that is taken to identify and collect system configurations. If identification and collection of system configurations were simplified, package design also were simplified. The patterns of identification and collection of system configurations are now enabled through a dedicated feature called *migration collection*.

2.2.1 Migration collection

A migration collection represents a group of system configurations that must be migrated from a source environment to a target environment.

The migration collection feature includes the following key characteristics:

- A migration collection is defined to hold one or more system configurations.
- Upon definition, system configurations are added to the migration collection in a number of ways over time.
- System configurations in the migration collection can be validated for completeness.
- When a migration collection is considered to be complete, a migration package definition can be generated from the migration collection and its contents.

A migration package that is created from a collection is different from a snapshot or a change package. The type of this package is called *collection* and is comprised solely of the system configurations that are included in the migration collection at the time the package was created.

Migration collections offer a simplified approach to identifying and grouping system configurations for migration. Unlike a snapshot package that requires significant data model and SQL expertise or a change package that requires controlling precisely the system configuration changes to be made, creating and managing a migration collection is a much simpler task that does not involve the use of the Migration Manager application. Migration collections decouple the implementation effort from the migration effort. Effective usage of migration collection can ensure traceability in the development environment activity and increase the velocity of the migration. This approach is called a *collection-based migration*.

In Tivoli's process automation engine 7.5 release, Migration collections is an application under the Migration module.

For more information about migration collections and usage models, see Chapter 3, “Migration collections” on page 49.

2.2.2 Comparison

A comparison is a set of system-generated records in a particular product environment that represents the differences in system configurations metadata regarding another product environment. The system configurations that correspond to these differences can be collected into a migration collection and packaged for promotion to the second product environment.

The comparison feature includes the following key characteristics:

- Comparison performs a database comparison between the source and target database and lists the differences for review.
- Comparison requires and is driven by a package definition.
- Comparison runs as a background task and must run to completion.
- Comparison results are persisted in the database and can be reported on.
- Comparison is not limited to differences. In addition to the differences, it identifies matches and system configurations that are present in one product environment but not in the other.

The Comparison tool also offers a simplified approach to identifying and grouping system configurations for migration. Unlike a migration collection that is built up over time, comparison results can be immediately added into a migration collection and packaged for migration. This approach, called a *comparison-based migration*, meets a requirement that calls for system configurations in participating product environments to be identical.

In Tivoli's process automation engine 7.5 release, Comparison is a tab in the Migration Manager application.

For more information about Comparison and usage models, see Chapter 4, “Migration Manager Comparison” on page 75.

2.3 Deployment management features

Package deployment into a product environment was a discrete task that was started from the Migration Manager application. Upon starting, the system configurations in the package are imported into the product environment subject to data validation rules. After it is begun, package deployment runs to completion or stops upon encountering an error.

An implementer must plan for the following key aspects of deployment in addition to being familiar with and performing the deployment task. These aspects directly affect production rollout and the timely availability of business applications to the users of the business:

- Replacing values in the system configurations to meet the needs of the target environment. For example, replacing organization and site references of a source environment on the appropriate system configurations of the package.
- Reviewing, correcting, and recovering from errors that are encountered during the package deployment. For example, recovering from an error that results when a workflow process system configuration references an inactive communication template.
- Reviewing and resolving the products and version differences in the participating product environments. For example, a source environment can have an older Fix Pack of a product applied whereas a target environment can have a new Fix Pack.

Older releases of Tivoli's process automation engine did not offer any tools to support these aspects that affect package deployment. This lack of tools resulted in a significant disadvantage to implementers who employed workarounds or external utilities to resolve issues that pertain to value replacement, deployment errors, or product and version compatibility.

2.3.1 Value replacement

In Tivoli's process automation engine 7.5 release, the Migration Manager application offers a feature that enables implementers to define rules to replace values in system configurations. These rules are defined and enabled in the target environment that are based on the characteristics of the system configurations in that environment. After the rules are enabled, they are applied automatically by the Migration Manager to packages that are being deployed into the target environment.

Value replacement rules streamline the migration effort by providing implementers the ability to adapt system configurations to suit the target environment. Manual editing of a Migration Manager package XML document can be eliminated, which saves implementers time.

For more information, see 11.5.2, “Replacement rules” on page 297.

2.3.2 Deployment errors correction and recovery

In Tivoli's process automation engine 7.5 release, the Migration Manager application’s error management capability is enhanced to support error correction and recovery. This enhancement enables implementers to go beyond reviewing the error message to determine the nature of a deployment error. If the deployment error is the result of a data validation failure on a system configuration in the package, the Migration Manager application can be used to quickly correct the system configuration and continue the deployment of the package.

Error correction and recovery can streamline the migration effort by enabling implementers to minimize the need to re-create packages to recover from a data validation failure. Deployment can be paused while the error is reviewed and corrected and then resumed to continue on to complete the deployment.

For more information, see 15.10, “Error correction and package reprocessing” on page 400.

2.3.3 Product compatibility

Migration Manager’s deployment task includes checks that compare the product and version information of the source environment with the information of the target environment. These checks are intended to identify differences in the list of installed products and Fix Pack and Hot Fix levels. Having different products, Fix Packs, and Hot Fixes implies that the code base (and therefore the business rules) is different between the source and target environments. The Migration Manager’s product compatibility user interface assists an implementer in determining whether package deployment is feasible and warranted.

In Tivoli's process automation engine V7.5, the Migration Manager’s product compatibility user interface was improved to present product lists, Fix Pack, and Hot Fixes differences in a more intuitive manner. There also is an option to exclude extra products that are listed from the source environment to continue deployment.

2.4 Usability improvements

Migration Manager functionality is presented and exercised through a web-based user interface. Migration Manager tasks are performed in a sequential manner by using various user interface windows. The following improvements were incorporated in Tivoli's process automation engine 7.5 release to simplify the user interaction with the Migration Manager application:

- A system property to set a package definition to an approved status upon first save. Turning on this system property eliminates the need to start a change status action for a package definition in the Migration Manager application.
- Snapshot package definitions no longer require a separate activation step.
- A package distribution record is automatically generated whenever a package is distributed to a chosen target.
- When package deployment is initiated, the eSignature dialog is no longer presented. The eSignature option continues to be available in the product environment but it is not enabled by default.
- The following new or improved Migration Manager object structures are included:
  - DMMAXOBJECTCFG object structure now supports migration across relational databases. For example, a package that contains a business object definition that was created in a product environment that is running Oracle database can be migrated to a product environment that is running IBM DB2® database.
  - DMQEUERY object structure enables the migration of application queries.
  - DMINTERACTION object structure enables the migration of web service interactions.
  - DMSCRIPT and DMLAUNCHPOINT object structures enable the migration of automation scripts and associated launchpoint.
  - DMWSREGISTRY object structure enables the migration of web services.

2.5 Migration Manager new and enhanced functions summary

Following the description of new and enhanced functions in “Accelerated package design and creation features” on page 40, “Deployment management features” on page 42, and “Usability improvements” on page 45, the strategy for Migration Manager becomes evident. Migration Manager was expanded to offer a sophisticated set of functions from which the most appropriate functions can be adopted to run migration efficiently. Implementers can choose the functions that best suit the development cycle and the skills of team members that are tasked with system configuration collection and package management.

Following Tivoli's process automation engine 7.5 release, Migration Manager offers the following sets of functions:

- System configuration collection functions
- Package management functions
- Content management functions

The system configuration collection functions (primarily in the form of migration collections and comparison) are designed to identify the appropriate system configurations in the quickest and most complete manner possible, which leads to more robust packages. These functions are overlaid on the existing package and content management functions, which raises the overall value and usability of Migration Manager. The package management functions were enhanced to support streamlined deployment.

Figure 2-2 shows the sets of functions and how migration collections and comparison augment basic package management.

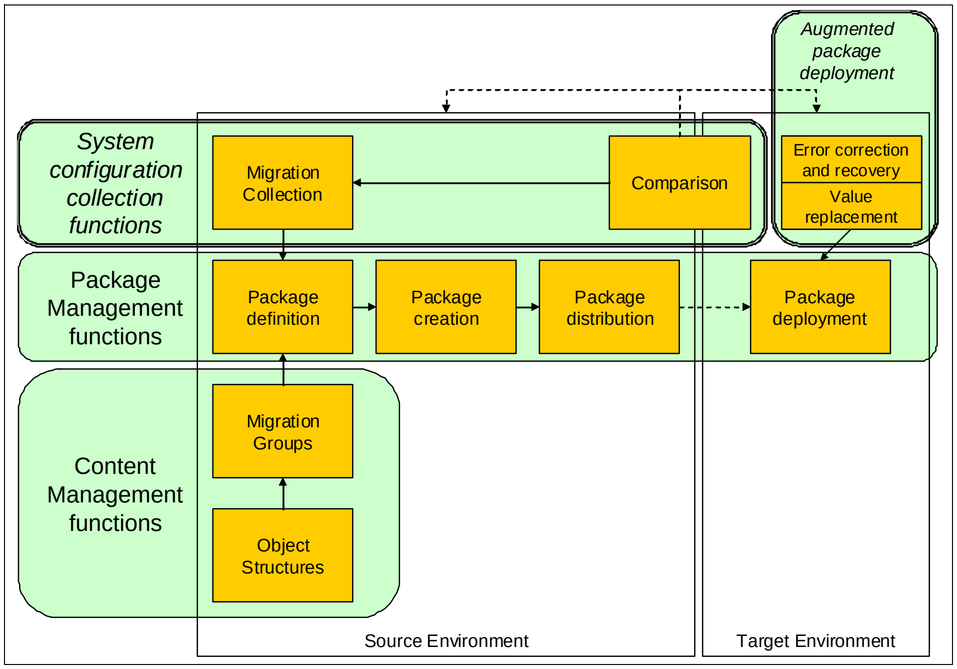


Figure 2-2 Migration collections and comparison augment basic package management

Subsequent chapters in this book describe the new and enhanced functions to achieve migration to production.

3

Migration collections

This chapter provides an overview on what collections are and how you can use them to simplify the migration package definition process.

In this chapter, we introduce the concept of collections and how by using collections, you can improve your support of the development process of your Tivoli's process automation engine that is based applications (IBM SmartCloud Control Desk or IBM Maximo Asset Management).

In this chapter, you also learn how to define the collection and how to use the various ways to populate the collection. In subsequent chapters, specific scenarios demonstrate how to use each of these methods.

This chapter includes the following sections:

- Migration collections strategy
- Collection stages
- Migration collections that are defined
- Collection validation
- Package definitions

3.1 Migration collections strategy

A migration collection is list of system configurations that must be packaged for promotion to upper environments. One or more collections typically come into existence in a development environment as a result of implementation tasks carried out by implementers or developers. Each collection is prepared by an implementer and contains records that are pointers to the actual system configuration records. At a specific point in the implementation cycle, the collection is processed into a Migration Manager package definition that starts the migration process.

3.1.1 Approach to collections

The nature of a collection is dependent upon the implementation approach that is used for the product. A collection always contains one or more system configurations that represent some aspect of implementation. The number and types of system configurations in the collection depend upon how the product implementation is done and the skills of the implementers.

Applications approach

The application approach is the most common approach the authors encounter. With the application approach, fully functional applications are rolled out to production users. Such applications are enabled by using a full complement of system configurations, including business objects, domains, validation scripts, user interface presentations, workflows, actions, and reports. All of the application functionality that is associated with the business application are placed into a self-contained collection or a series of collections.

This approach is useful when different implementers are tasked with addressing different applications of the product and possess the skills to create different system configurations.

System configurations approach

With the system configuration approach, a single implementer is tasked with preparing certain system configurations across multiple business applications due to expertise with the particular system configuration. For example, business object definitions for multiple business applications are managed by the implementer with expertise in Database Configuration. Each collection contains similar system configurations across multiple business configurations. For example, a data dictionary collection contains only business object and domain definitions.

With this approach, care must be taken to sequence the migration of the packages into a target environment. Data dictionary packages must be migrated before the application packages.

It is possible that as implementation decisions are made and priorities change, the nature of collections also change from one approach to the other. Migration Collections functionality does not mandate a rigid “one or the other” approach.

3.1.2 Benefits of the collections strategy

A collections-based migration strategy brings the following benefits:

- Align better with the implementation effort and the tasks that are run

  By using collections, the implementer can incrementally identify and select the result of implementation tasks. The collection can grow or shrink over time, which more accurately reflects the implementation. Multiple intuitive options are available to populate a collection.

- Reduce the need for expertise on application data model

  Collections are aligned with applications. Collections are populated over time by accessing the applications and identifying the wanted system configuration. Unlike Migration Manager snapshot mode, there are no SQL criteria to specify; therefore, significant SQL knowledge is not required to collect a system configuration.

- Related data can be identified more quickly

  At any time, the entries in a collection can be validated to identify potential related system configurations that might be needed to complete migration. This validation can be done through the Migration Collections application as a background job. Validation scan results are displayed to the implementer, which enables further selection of wanted system configurations. This validation scan is much faster than attempting to collect every required system configuration, especially when changes are not well-documented.

- Collections are seamlessly integrated with Migration Manager

  Collections can be populated from scratch based on a Migration Manager-driven comparison with a remote product environment. This configuration is a powerful model that requires little set up on the part of the implementer. A Migration Manager package definition can be generated from a collection through a single application action.

- Collections offer more granular traceability

  When system configurations are placed into collections, there is a more granular traceability of the implementation effort. Collections can demarcate the set of changes to a particular business application, the specific part of the implementation cycle and the specific implementer that made the change.

- Collection functionality is easily extended

  Collections are aligned with the system applications that are used most often to implement system configurations. However, other applications also can be enabled to participate in collections if they have supporting object structures. By incorporating rules, more Related Data also can be found.

- Collections support a more granular version control model

  A collection can be populated by using XML-based system configuration representations that are managed in a version control system. From a collection, XML-based representation of a system configuration can be exported and saved for check-in to a version control system. Such a granular model of managing individual system configurations cannot be implemented with Migration Manager packages.

3.2 Collection stages

A collection can be created at any time. Upon creation, it has a name and description only. From that point onward, system configuration entries can be added to or removed from the collection. Entries in the collection can be validated to identify Related Data. Such Related Data can be added into the collection. When the collection is considered to be complete, a Migration Manager package is created from the collection. The stages of a collection are shown in Figure 3-1.

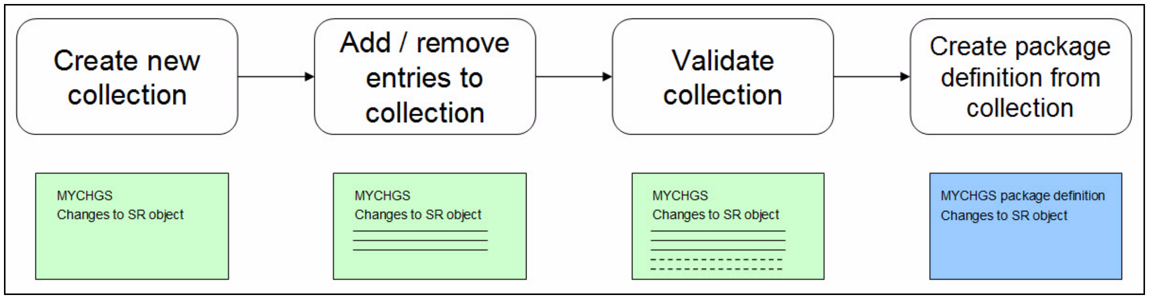


Figure 3-1 Stages in a migration collection

A collection also can be deleted at any time. However, if packages are defined based on the collection, we do not recommend deleting the collection. Doing so removes an essential link in the chain of traceability of system configurations.

Important: The following key aspects of a collection can help streamline a collection strategy to migration:

- A collection cannot be duplicated or shared in any form. A collection is designed for a particular product environment to represent a list of system configurations that are implemented in that product environment.
- A collection cannot be combined or merged with other collections.
- One collection implies one migration package, except when the collection contains certain circular dependencies. If certain circular dependencies exist, the collection is split into one or more migration packages.
- A collection facilitates package management and packages alone are migrated between environments. A collection should be perceived merely as a data collection construct.

3.3 Migration collections that are defined

Defining migrations collections is accomplished first by entering a new record in the Migration Collections application as with any other Tivoli's process automation engine application. Enter a key value and a description and save the record. You can decide whether to make this Collection public or not by selecting the **Public?** option, which makes it available to everyone.

3.3.1 What is a collection entry?

A collection includes one or more entries called Configurations. Each such Configuration entry is not the actual system configuration itself but a pointer to the main system configuration record. An identifier for that system configuration record, its primary key columns and values, and the application for the system configuration (if applicable) are stored as the Configuration entry of the collection. The implementer can follow the link to the next application name to view or edit the system configuration record in its respective application.

3.3.2 Options for populating a collection

There are multiple options to add entries into a collection. Each option reflects a common pattern of product usage or data collection technique that is prevalent among implementers and system administrators. Any combination of these options can be used to assemble the wanted collection. Table 3-1 lists the options and describes the behavior that is associated with each option.

Table 3-1 Options to populate a collection

Option	**Usage and benefits**
Go to → Return with value	This option is used to browse to a chosen system configuration application and return a single value to the collection. The collection must exist before this option is used. The option is available only for primary system configurations that have a dedicated application.
Migration collection toolbar button	This option uses an "Add to migration collection" toolbar button that dynamically adds to the wanted system configuration application. The collection must exist before the button can be used to select and add system configuration. One or multiple entries can be selected from the list tab of the application for inclusion into a chosen collection. With this option, a collection can be populated without leaving the system configuration application that an implementer is using.
Event tracking	This option enables a behind-the-scenes tracking of system configurations that can be added, updated, or deleted by implementers. The collection must exist before tracking can be enabled. This option precludes the need to manually select one or more system configurations. However, this option must be enabled before any system configurations are implemented. Event tracking is not performed when the product environment remains in Admin Mode.
Import XML	This option enables a model of importing system configurations in the form of XML data from external repositories, such as a version control system. The collection must exist before import can be used. Multiple configurations can be simultaneously imported into the product environment and added into the collection. Collection must exist before import is initiated.

Option	Usage and benefits
Comparison	This option enables a collection to be created and populated with the results of a comparison between two product environments. Only system configurations that are identified by comparison to be local only and difference can be added to the collection.
Validation	This option requires the implementer to first run the Validate Migration Collection action. The results of validation are reviewed and the appropriate related system configurations are added into the collection. The collection must exist and contain at least one entry before validation can be run. Multiple related system configurations can be selected from validation results and added into the collection.

Subsequent chapters in this publication use these options to prepare collections of various system configurations.

3.4 Collection validation

Collection validation is a mechanism to uncover related system configurations in a product environment and list them for review and potential inclusion in an existing collection. Validation takes the form of a scan of system configuration data that is based on rules. These rules are configurable by an administrator or implementer. There are multiple outcomes from running a validation against a collection. The most predominant and relevant outcome is a list of related system configurations that should be reviewed by the implementer. These configurations also should be considered for inclusion in the collection, which brings the collection to a complete stage.

Although collection validation is not mandated, running validation is beneficial in that it shifts the burden of identifying related system configurations from an implementer to the system. Not only does this shift lower the bar on migration skills, but also leads to complete migration packages that are designed and delivered in less time.

Validation runs as a background job in the product environment. The time that is taken to complete validation depends upon the types and numbers of system configurations in the collection and the validation rules in effect. After it is started, a validation background job cannot be canceled; it must run to completion.

Best practice: Run validation at least once against a collection before a package definition is created from the collection. Review validation results for potential system configurations to be added into the collection for completeness and to ensure that dependencies are met.

3.4.1 Validation results

After you create the collection, you want to validate that you collected all of the related configurations. You make this validation by using **Select Action** → **Validate Migration Collection**. Validation is manually initiated after a migration collection is created but before any package definitions are generated.

This process populates one or both of the two sections with validation results and related configurations. By using this information, you can determine whether more steps are needed to properly migrate all related configurations without migrating unnecessary duplicate data.

Important: Validation cannot run when the product environment's Administration Mode is set.

Validation process results

After the validation process completes, the results show in the top section of the page. You can perform actions on any orphans or duplicates. If any such records exist, any or none of the following actions can be taken:

- Delete orphans

  Orphans are entries in a collection whose corresponding system configuration records no longer exist in the product environment. Collection entries are not constantly synchronized with their corresponding system configuration records. If a system configuration is deleted, the collection entry is no longer linked to the system configuration main record and thus becomes an orphan. Orphans should be deleted to keep the collection current.

- Delete duplicates

  An entry in the collection is considered to be duplicate when the primary keys of the corresponding system configuration match those keys of another system configuration that also is included in the collection. This match can happen when a system configuration is created, included in the collection, then deleted, and re-created again with the same primary keys. Although the system configuration keys are the same, other values of the system configuration record are different. Duplicates should be deleted to keep the collection current.

- Delete all

  This action deletes orphans and duplicates.

Related configurations

Any related configurations appear in the Related Configurations section after the validation process completes. You must now decide which configurations are necessary for the migration to be successful. The migration of redundant data is now reduced because you can see the related configurations that are required for validation. If your target system has the data, you can delete the related validation entry. For those entries that are still needed, you can add those entries into the collection. You can perform actions on the related configurations by choosing one of the following options:

- Delete all related configurations entries
- Add all related configuration entries to migration collection
- Delete related configuration entry
- Add related configuration entry to migration collection

Each of these actions offer convenient options to add selected related configurations into the collection or discard related configurations from the Related Configuration section. By using these add and delete options, you can quickly reduce the number of related configurations to review.

3.4.2 Validation rules

The following types of validation rules are applied in sequence during the validation of a collection to uncover related configurations:

- Data dictionary metadata-based rules

  The product's data dictionary metadata can help identify related configurations. A business object's attribute metadata is used for this purpose. Data dictionary metadata rules cannot be defined or turned off, but they can be controlled by using system properties.

- Related Data rules

  Primary system configurations (such as business objects) can be directly selected by using the corresponding application, such as Database Configuration. However, secondary configurations (such as relationships) do not feature a separate application. Such secondary configurations are often managed through pop-up windows that are started from within a business application or tabs within the primary system configuration application. Such secondary configurations are identified by using Related Data rules. Related data rules can be enabled or disabled as appropriate. New rules also can be defined.

- Lookup rules

  A subset of system configurations does not include dedicated management applications. In addition, these system configurations have no relationships to other configurations. For example, messages that are used in a product environment are a stand-alone configuration. Such configurations are selected for inclusion in a collection by using lookup data rules. Lookup data rules can be enabled or disabled as appropriate. New rules also can be defined.

Important: Validation rules include the following important considerations:

- System configurations that are identified as a result of running lookup rules are added directly into the collection and not displayed in the Related Configurations section of Validation Results tab. These system configurations are not related to any primary system configuration that is already in the collection. Such system configurations should be removed directly from the collection if they are not required for the particular migration.
- Lookup rules are global to all collections. If one or more lookup rules are enabled, any collection that is validated has entries added in as a result of running the lookup rule. Implementers should turn off lookup rules when they are not needed or manually delete lookup result records from collections where the results are not required or appropriate.

3.4.3 Managing validation rules

Validation rules are a powerful construct that must be managed over time to ensure efficient and appropriate collection of system configurations. Effectively managing rules ensures that collections are well-designed and complete, which ensures accelerated migration.

Data dictionary metadata rules

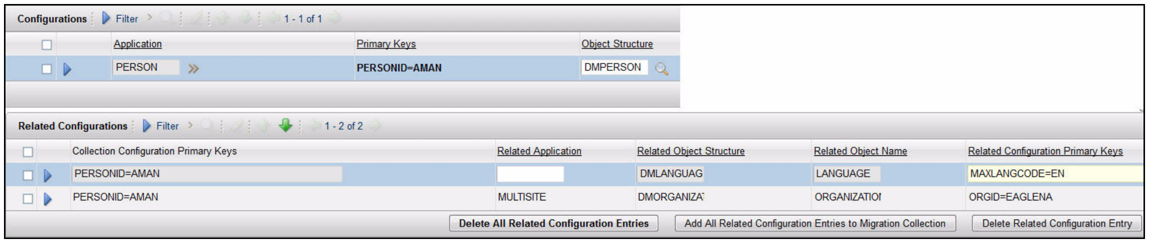
Data dictionary metadata rules are derived automatically by the collection validation function. The validation function identifies business objects and attributes that comprise a particular system configuration and builds paths to related business objects by using data dictionary metadata. These paths are traversed to determine whether a related system configuration record is present at a node on this path. The function then checks whether the primary system configuration record is dependent on the related system configuration record found. If so, the related system configuration record is added to the list of related system configurations.

The key concept to understand is that related system configurations are identified by traversing paths. Some paths might lead to business objects that are not relevant to a primary system configuration; however, they appear to be related purely by virtue of data dictionary metadata. Traversal of such paths should be eliminated to keep the validation function efficient.

Data dictionary metadata-based paths

We use an example to show how data dictionary metadata rules are synthesized by the validation function.

In a product environment that has demonstration data, a collection is created and a PERSON record is added to the collection. Validation is run on the collection. Validation Results show two related records: a LANGUAGE record and an ORGANIZATION record, as shown in Figure 3-2.


Configurations
Filter
1 - 1 of 1

Application	Primary Keys	Object Structure
PERSON	PERSONID=AMAN	DMPERSON

Related Configurations
Filter
1 - 2 of 2

Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
PERSONID=AMAN		DMLANGUAG	LANGUAGE	MAXLANGCODE=EN
PERSONID=AMAN	MULTISITE	DMORGANIZAT	ORGANIZATIOI	ORGID=EAGLENA

Delete All Related Configuration Entries
Add All Related Configuration Entries to Migration Collection
Delete Related Configuration Entry

Figure 3-2 Collection entry and related configurations

The related system configurations LANGUAGE and ORGANIZATION were identified based on data dictionary metadata that is associated with the PERSON business object and its child objects, such as LONGDESCRIPTION, EMAIL, SMS, PHONE, and PERSONCAL. Validation functionality determined that there are two potential paths to follow that could identify related system configurations for the PERSON record. A PERSON record could be associated with ORGANIZATION (through attributes LOCATIONORG or PRIMARYCALORG) and LANGUAGE (through attributes LANGCODE).

From a migration perspective, language and organization records are migrated before people records are moved. Therefore, the language and the organization that is referenced on the person record are deemed to be related system configurations that the PERSON record depends upon. Figure 3-3 on page 60 shows the paths that are followed by the validation functionality and the criteria that drives the decision to follow the path.

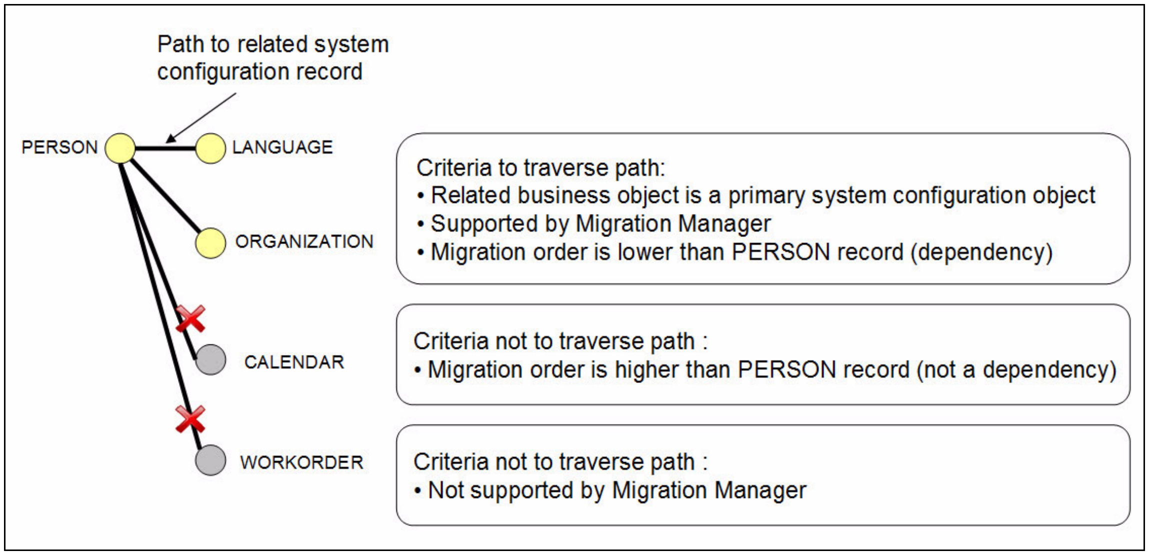
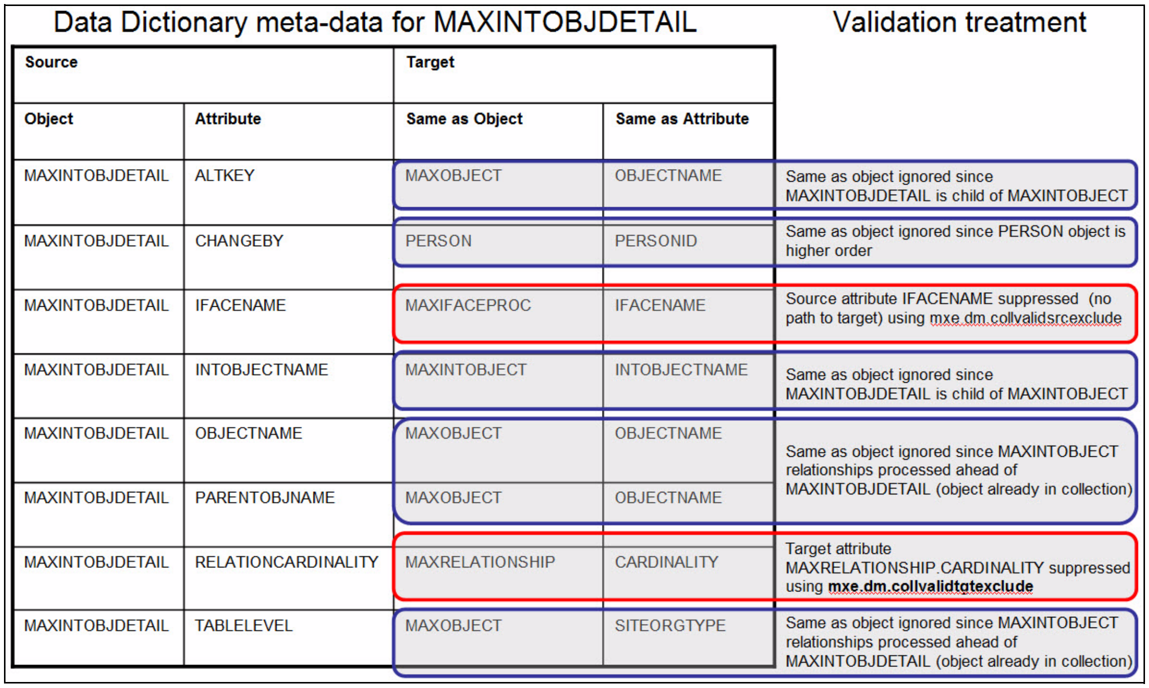

Path to related system configuration record
PERSON
LANGUAGE
ORGANIZATION
CALENDAR
WORKORDER
Criteria to traverse path:
• Related business object is a primary system configuration object
• Supported by Migration Manager
• Migration order is lower than PERSON record (dependency)
Criteria not to traverse path :
• Migration order is higher than PERSON record (not a dependency)
Criteria not to traverse path :
• Not supported by Migration Manager

Figure 3-3 Traversing paths to select related system configurations

Figure 3-3 also shows the paths that would not be followed even though the data dictionary metadata may identify other related objects. The path to the CALENDAR record is not followed even though calendar configurations are supported by Migration Manager because calendars have a migration order higher than the person configuration. The path to the WORKORDER record is not followed because work order migration is not supported by Migration Manager by default.

Sources and targets

A path that is defined by the validation function can be considered to be built from a series of nodes. Each node represents a business object. When two business objects lie along a path, the source object is where the path begins from and the target object is to where the path points. The data dictionary's "same as" declarations are used by the validation function to identify sources and targets. Figure 3-4 uses the MAXINTOBJDETAIL object and its defined same-as relationship metadata to show sources and targets. Also shown is how the validation function treats each attribute of MAXINTOBJDETAIL to determine whether paths must be built to find related system configurations.



Data Dictionary meta-data for MAXINTOBJDETAIL

Source		Target		Validation treatment
Object	**Attribute**	**Same as Object**	**Same as Attribute**	
MAXINTOBJDETAIL	ALTKEY	MAXOBJECT	OBJECTNAME	Same as object ignored since MAXINTOBJDETAIL is child of MAXINTOBJECT
MAXINTOBJDETAIL	CHANGEBY	PERSON	PERSONID	Same as object ignored since PERSON object is higher order
MAXINTOBJDETAIL	IFACENAME	MAXIFACEPROC	IFACENAME	Source attribute IFACENAME suppressed (no path to target) using mxe.dm.collvalidsrcexclude
MAXINTOBJDETAIL	INTOBJECTNAME	MAXINTOBJECT	INTOBJECTNAME	Same as object ignored since MAXINTOBJDETAIL is child of MAXINTOBJECT
MAXINTOBJDETAIL	OBJECTNAME	MAXOBJECT	OBJECTNAME	Same as object ignored since MAXINTOBJECT relationships processed ahead of MAXINTOBJDETAIL (object already in collection)
MAXINTOBJDETAIL	PARENTOBJNAME	MAXOBJECT	OBJECTNAME	
MAXINTOBJDETAIL	RELATIONCARDINALITY	MAXRELATIONSHIP	CARDINALITY	Target attribute MAXRELATIONSHIP.CARDINALITY suppressed using **mxe.dm.collvalidtgtexclude**
MAXINTOBJDETAIL	TABLELEVEL	MAXOBJECT	SITEORGTYPE	Same as object ignored since MAXINTOBJECT relationships processed ahead of MAXINTOBJDETAIL (object already in collection)

Figure 3-4 Same as relationship declarations and their use in validation

In Figure 3-4, two same-as attributes must be processed differently by the validation function. Paths that are defined by using these same-as attributes do not yield useful validation results. In the next section, we walk through the system properties that control the paths that are defined by the validation function.

Important: The terms target and source should not be confused with target and source for package definition. Those terms relate to system level processing of data from one environment to the next. Source and target here refer to the validation process of source `object.attribute` to target `object.attribute` by using the SAMEASOBJECT and SAMEASATTRIBUTE definitions to determine which objects require validation that is based on those values that are found in the product environment's data dictionary metadata.

System properties that are controlling metadata paths

Three system properties control how paths are defined and traversed. The use of each property is shown with an example in the following sections.

Table 3-2 System properties that are controlling paths

Property name	Usage
mxe.dm.collvalidlevels	The value for this system property determines the depth of the path that is followed by the validation function when related system configurations are uncovered.
mxe.dm.collvalidsrcexclude	The value for this system property is a tokenized list of object, attributes, and attribute value combinations that should be not be traversed from when a path is defined.
mxe.dm.collvalidtgtexclude	The value for this system property is a comma-separated list of objects and attributes combinations that should not be traversed to during validation.

System property mxe.dm.collvalidlevels

Figure 3-5 shows the effect of setting system property mxe.dm.collvalidlevels. When the level is set to 1, the validation function traverses a path from a communication template system configuration to its creator person record; in other words, from COMMTEMPLATE object to PERSON object. When the level is set to 2, the validation function traverses a longer path from COMMTEMPLATE object to PERSON object to ORGANIZATION object.

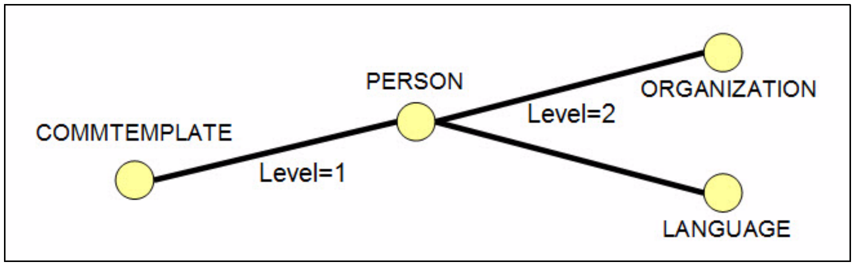


Figure 3-5 Effect of changing validation levels

A level of 1 or 2 is sufficient to identify the predominance of the system configurations that must be migrated.

System property mxe.dm.collvalidsrcexcclude

The system property mxe.dm.collvalidsrcexclude can eliminate certain paths that the validation function can traverse based on the metadata that is declared in the data dictionary. Traversing these paths might not yield the correct results for validation.

The system property is so named because the values that are provided reflect source objects. A path connects a source object to a target object. By specifying values for this system property, we ensure path traversal ends when a certain source object and its attributes are found during the validation scan. The validation function does not attempt to build a path from the source to target.

Figure 3-6 on page 64 shows when this property can be used to teach the validation function not to traverse unwanted paths.

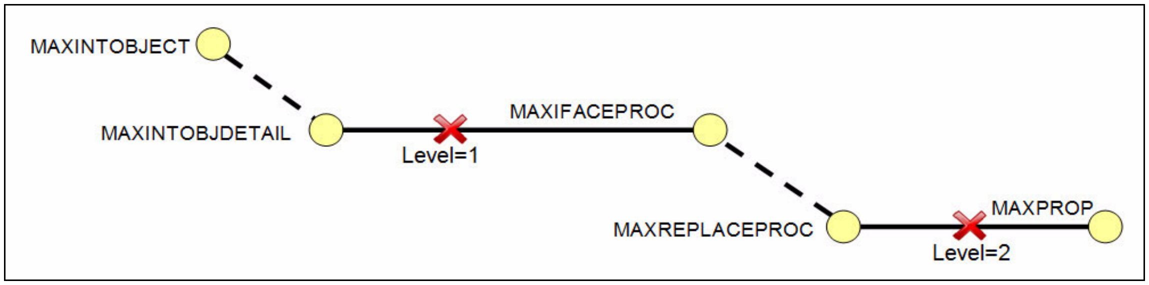


Figure 3-6 Excluding certain paths by using mxe.dm.collvalidsrcexclude property

In Figure 3-5, an object structure (MAXINTOBJECT), for instance, MXPO, is added to the collection. Validation level is set to 2 and validation is started. Validation results are returned that include the business objects that are used by the object structure and a Mapping Rules (MAXIFACEPROC) and System Properties (MAXPROP). Neither of these results is appropriate because the goal for this scenario is to prepare an object structure collection and its truly related configurations. An object structure does not directly rely upon Mapping Rules or System Properties to be functional.

If validation level is set to 1 and the validation run, none of the system properties are returned because the validation function traverses a shorter path (only those business objects that are directly related to MAXINOBJECT). However, the path from MAXINTOBJDETAIL to MAXIFACEPROC also should be eliminated to ensure that only relevant results are returned. By adding the value <MAXINTOBJDETAIL.IFACENAME> to the mxe.dm.collvalidsrcexclude system property, we are effectively removing the path that the validation function follows. Running validation once again now yields precisely the results that we expect to see.

System property mxe.dm.collvalidtgtexclude

The system property mxe.dm.collvalidsrcexclude can eliminate certain objects and attributes that the validation function might find as part of the metadata that is declared in the data dictionary. The system property is so named because the values that are provided reflect target objects. Because a path connects a source object to a target object, this property represents objects and attributes that appear as targets. While a valid path can be traversed from a source to the target, building the next path from the specific target object might yield irrelevant results.

Figure 3-7 shows when this property can be used to teach the validation function not to build unwanted paths.

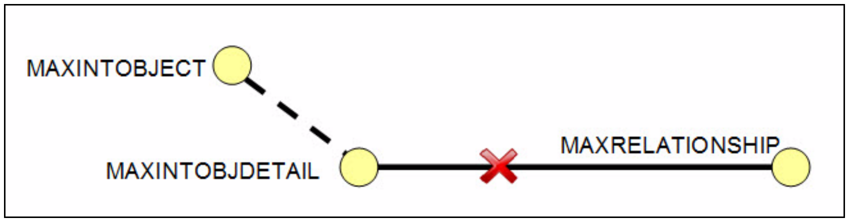


Figure 3-7 Excluding objects by using mxe.dm.collvalidtgtexclude property

In Figure 3-7, MXPO object structure is added to the collection. Validation yields a significant number of relationships (MAXRELATIONSHIP). Many of these relationships are for objects that have nothing to do with PO. Some of the relationships are defined to support reports. These results are irrelevant to the object structure collection.

In the data dictionary, the MAXINTOBJDETAIL object includes defined same-as relationships to the MAXRELATIONSHIP object. The source is RELATIONCARDINALITY attribute, target is MAXRELATIONSHIP.CARDINALITY. While we do want to retrieve value for the source RELATIONCARDINALITY, we also do not want to pursue the path to the target MAXRELATIONSHIP.CARDINALITY. Thus, the target object and attribute are added into the mxe.dm.collvalidtgtexclude system property.

Defining Related Data rules

Secondary system configurations that do not have dedicated management applications can be identified and collected during validation by defining and enabling Related Data rules. Such rules also can be disabled, which limits the number of secondary system configurations that are identified through validation.

Related Data rules are defined, enabled, or disabled by using **Select Action** → **Define Related Data and Lookup Rules**, as shown in Figure 3-8. This menu (select **Action** → **Define Related Data and Lookup Rules**) gives you the option of determining the data to exclude in the validation process. Figure 3-9 on page 67 shows the Lookup Rules tab in this dialog box. For more information about the lookup tab, see , “Defining Lookup rules”.

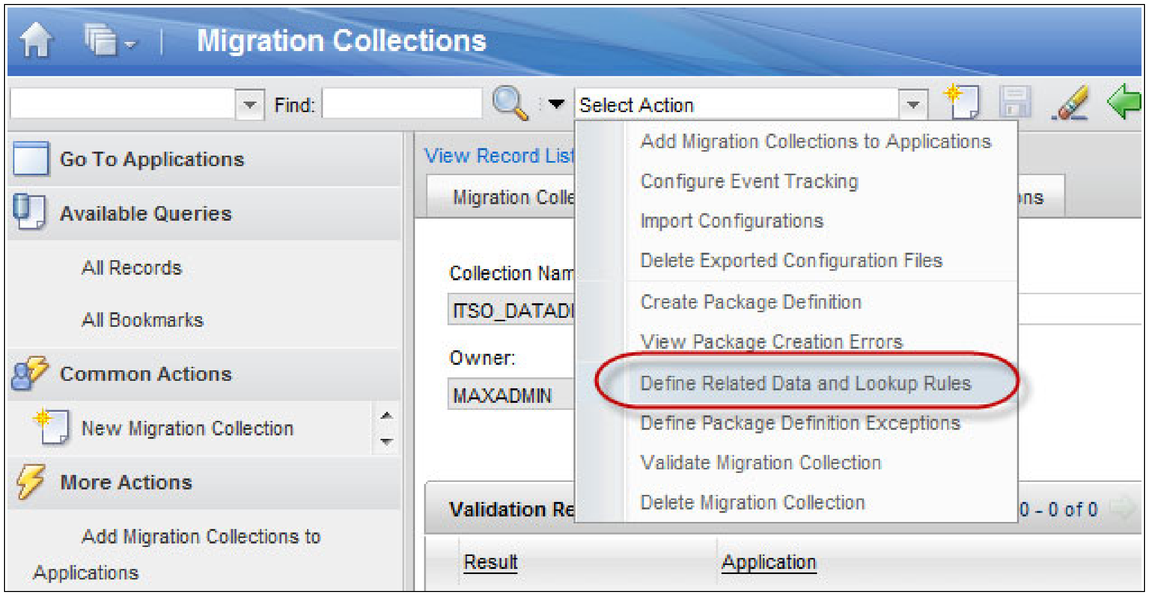
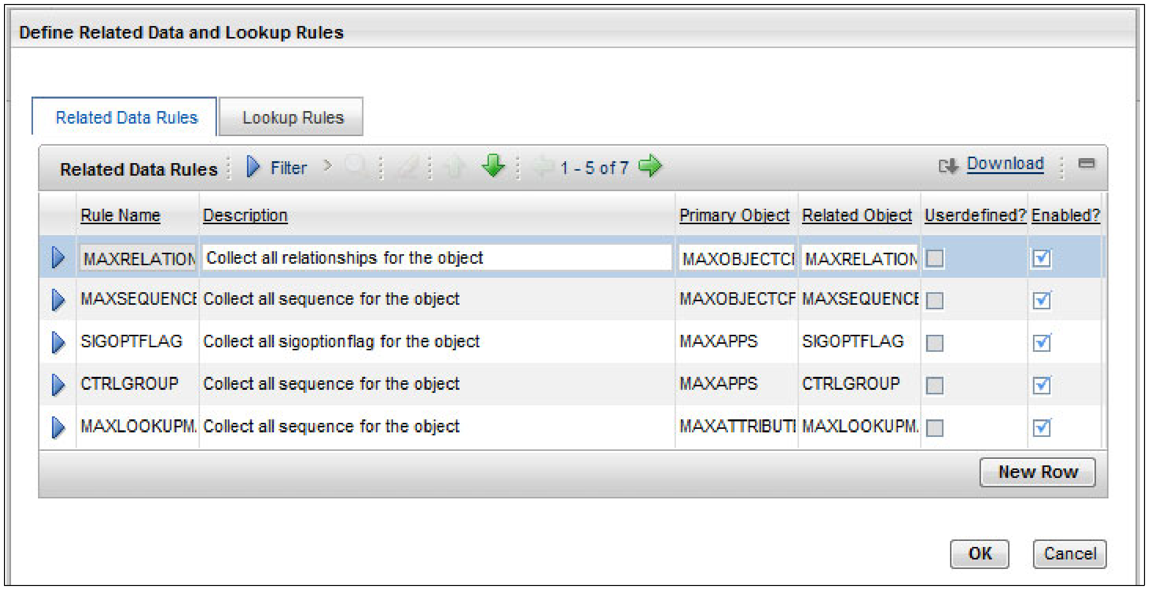


Figure 3-8 Define Related Data and Lookup Rules

Related Data rules govern how collections search the database for related configurations for those configurations that are listed in the collection. There are several rules for Related Data and lookups. Figure 3-9 shows the Related Data rules that are provided.


Define Related Data and Lookup Rules

Related Data Rules | Lookup Rules

Related Data Rules — 1 - 5 of 7

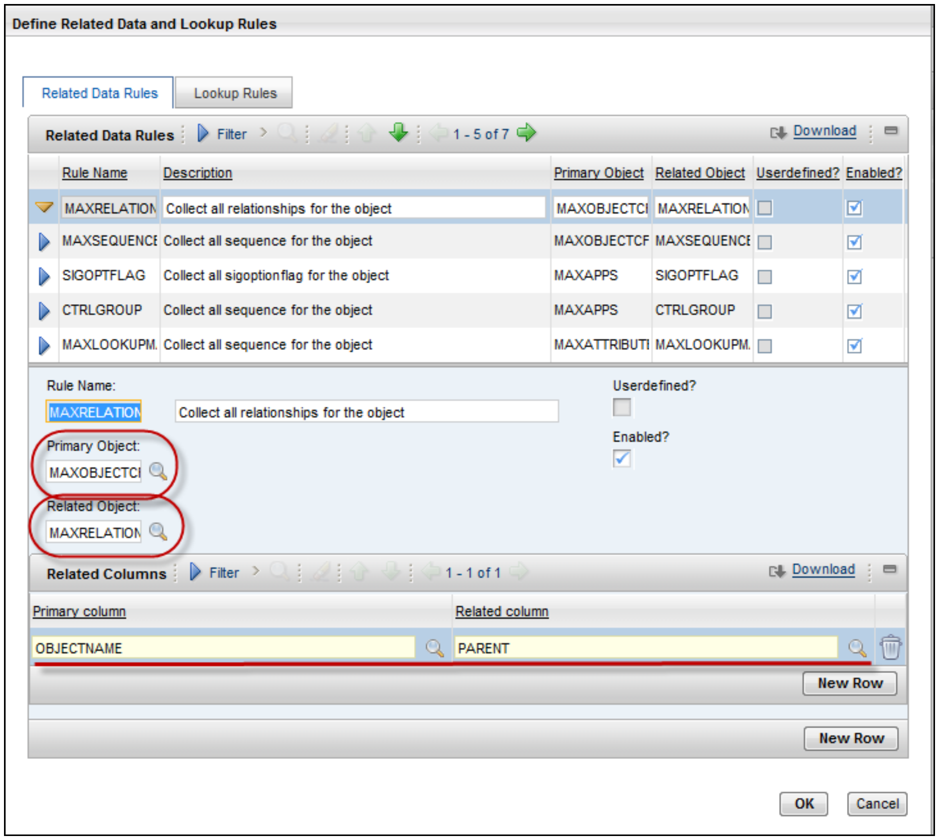
Rule Name	Description	Primary Object	Related Object	Userdefined?	Enabled?
MAXRELATION	Collect all relationships for the object	MAXOBJECTCI	MAXRELATION	☐	✓
MAXSEQUENCE	Collect all sequence for the object	MAXOBJECTCF	MAXSEQUENCE	☐	✓
SIGOPTFLAG	Collect all sigoptionflag for the object	MAXAPPS	SIGOPTFLAG	☐	✓
CTRLGROUP	Collect all sequence for the object	MAXAPPS	CTRLGROUP	☐	✓
MAXLOOKUPM.	Collect all sequence for the object	MAXATTRIBUTI	MAXLOOKUPM.	☐	✓

Figure 3-9 Related data rules provided

These rules provide the basis for getting all of the configurations that are necessary to ensure that the changes that were made during development are properly captured in the package definition.

In the following example, we describe how the Related Data rules work.

The first rule that we examine is the MAXRELATIONSHIP rule. This rules states that all relationships for an object are collected. The Primary Object is the object that is defined as the Main table in the configuration application for which an object structure is defined and is used by the Migration Manager application. The Related object is the object that contains the related configurations and is not a main table object in the previously mentioned Object Structures. Figure 3-10 on page 68 shows how this Related object is created. In Figure 3-9, the column definitions that link the two objects also are defined. This definition could include multiple columns.


Define Related Data and Lookup Rules

Related Data Rules | Lookup Rules

Related Data Rules — 1 - 5 of 7

Rule Name	Description	Primary Object	Related Object	Userdefined?	Enabled?
MAXRELATION	Collect all relationships for the object	MAXOBJECTCI	MAXRELATION	☐	✓
MAXSEQUENCE	Collect all sequence for the object	MAXOBJECTCF	MAXSEQUENCE	☐	✓
SIGOPTFLAG	Collect all sigoptionflag for the object	MAXAPPS	SIGOPTFLAG	☐	✓
CTRLGROUP	Collect all sequence for the object	MAXAPPS	CTRLGROUP	☐	✓
MAXLOOKUPM.	Collect all sequence for the object	MAXATTRIBUTI	MAXLOOKUPM.	☐	✓

Rule Name: MAXRELATION — Collect all relationships for the object

Userdefined? ☐

Enabled? ✓

Primary Object: MAXOBJECTCI

Related Object: MAXRELATION

Related Columns — 1 - 1 of 1

Primary column	Related column
OBJECTNAME	PARENT

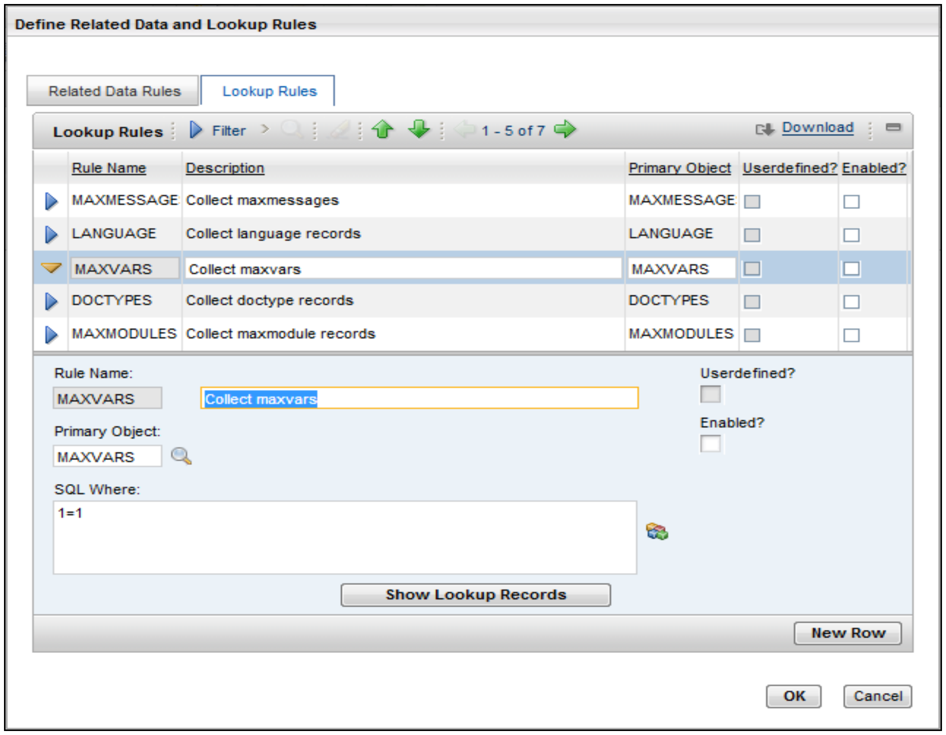
Figure 3-10 Related data rule defined

In this example, you can add other business relationships for your configurations to add or not add particular configurations as your system requires.

Defining Lookup rules

Certain system configurations do not have a dedicated management application nor relationship to other system configurations. Identification and selection of such configurations are performed by using Lookup rules.

In Figure 3-11 you see that the MAXVARS object is a look-up. The rule is to collect all MAXVARS. This rule can be changed per your implementation requirements to exclude all (for example 1=2) or to limit the record selection to a pertinent record set for your business requirements. So, if you are adding an SITEID, you could modify the SQL Where clause to limit the records only to your new SITEID.



Rule Name	Description	Primary Object	Userdefined?	Enabled?
MAXMESSAGE	Collect maxmessages	MAXMESSAGE	☐	☐
LANGUAGE	Collect language records	LANGUAGE	☐	☐
MAXVARS	Collect maxvars	MAXVARS	☐	☐
DOCTYPES	Collect doctype records	DOCTYPES	☐	☐
MAXMODULES	Collect maxmodule records	MAXMODULES	☐	☐

Rule Name: MAXVARS Collect maxvars

Userdefined? ☐

Primary Object: MAXVARS

Enabled? ☐

SQL Where: 1=1

Figure 3-11 Related lookup rules

Important: You can gain a significant understanding of the validation function by reviewing the product log files. To ensure detailed log statements are written to the product log, ensure the Migration Manager logger `dm` is set to DEBUG level and an appender is configured to write the log statements to.

Validation logs begin with a statement similar to the following example:

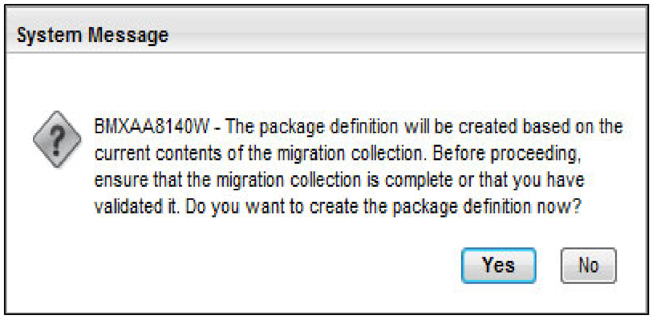
```
*************Validation started for collection= 'REDPATH1'  and
source= 'ibm-rhkb7omky1r_MX7501_MAXIMO'
```

Validation logs end with a statement similar to the following example:

```
*************Validation complete for collection= 'REDPATH1'  and
source= 'ibm-rhkb7omky1r_MX7501_MAXIMO'
```

3.5 Package definitions

When you are satisfied with the configuration entries for your collection, you can generate the package definition that represents the collection. Click **Select Action** → **Create Package Definition** from the Select Action menu. You see the dialog box that is shown in Figure 3-12.



BMXAA8140W - The package definition will be created based on the current contents of the migration collection. Before proceeding, ensure that the migration collection is complete or that you have validated it. Do you want to create the package definition now?

Figure 3-12 System message for create package definition action

This dialog box gives you an opportunity to back out of generating the package definition in case you must review the collection.

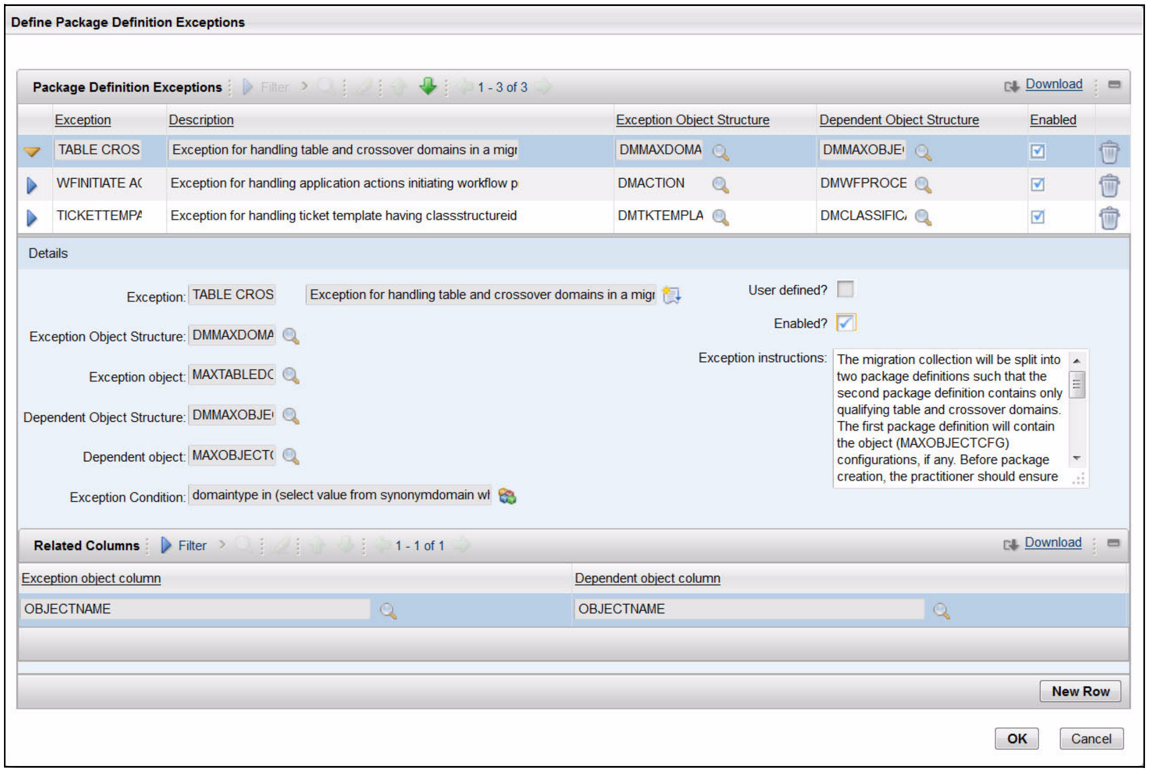
The package definition generation from a collection can be perceived as the reversal and automation of the typical migration approach. In this approach, a package definition is manually defined, migration groups are included, and SQL criteria are defined to narrow down the system configurations to be included in the package. Because a collection is finalized, the records that must be packaged are now known up front. The collection functionality is able to determine which migration groups the collection entries belong to and assembles the migration groups and the package definition to contain the system configurations.

3.5.1 Package definition exceptions

There are situations where system configurations might depend upon one another in a circular manner. For example, a table or crossover domain might use an attribute where the attribute points back to the domain. These situations, which we refer to as chicken-and-egg, are addressed in collections by using package definition exceptions.

Package definition exceptions are rules that define circular relationships and trigger the splitting of a collection into more than one package definition. This configuration is done to facilitate a more controlled and sequential migration of system configuration and prevent data validation failures in the target product environment. Essentially, the circular relationship is severed for migration.

Figure 3-13 shows the Define Package Definition Exceptions window that is used to define and manage exception rules. As shown in this figure, an exception rule TABLE CROSSOVER DOMAIN identifies the combination of business objects that constitute the circular dependency. If system configurations that are based on these objects are found in the collection, the collection must be split into more than one package definition. The rule definition also provides instructions to implementers on how to migrate the system configurations, including any manual activity to be undertaken that is not automated by Migration Manager.



Define Package Definition Exceptions

Package Definition Exceptions Filter 1 - 3 of 3 Download

Exception	Description	Exception Object Structure	Dependent Object Structure	Enabled
TABLE CROS	Exception for handling table and crossover domains in a migr	DMMAXDOMA	DMMAXOBJE	✓
WFINITIATE AC	Exception for handling application actions initiating workflow p	DMACTION	DMWFPROCE	✓
TICKETTEMPA	Exception for handling ticket template having classstructureid	DMTKTEMPLA	DMCLASSIFIC	✓

Details

Exception:	TABLE CROS — Exception for handling table and crossover domains in a migr
Exception Object Structure:	DMMAXDOMA
Exception object:	MAXTABLEDC
Dependent Object Structure:	DMMAXOBJE
Dependent object:	MAXOBJECT(
Exception Condition:	domaintype in (select value from synonymdomain wl
User defined?	☐
Enabled?	✓
Exception instructions:	The migration collection will be split into two package definitions such that the second package definition contains only qualifying table and crossover domains. The first package definition will contain the object (MAXOBJECTCFG) configurations, if any. Before package creation, the practitioner should ensure

Related Columns Filter 1 - 1 of 1 Download

Exception object column	Dependent object column
OBJECTNAME	OBJECTNAME

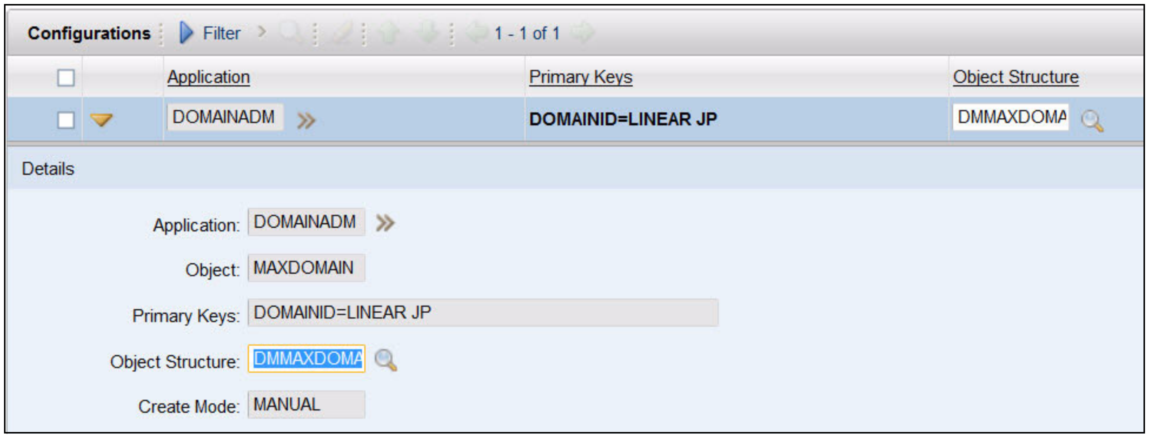
Figure 3-13   Package definition exception rule user interface

A few rules are supplied with the product and more rules can be defined as needed. For more information about the specific steps that are used to build rules, see the product documentation.

Package definition exceptions and package definition errors

If a collection contains system configuration entries that are affected by package definition exception rules, the collection functionality checks to see whether the dependent system configuration also is part of the collection. If it is not, package definition is not created and errors are recorded for review. For example, if a table domain is added to the collection and the Create Package Definition option is run, the definition is not created. The Package Definitions tab remains empty. If this behavior is observed, the View Package Creation Errors option should be run to review errors.

Figure 3-14 shows a table domain LINEAR JP that is the only entry in a collection. The collection is saved and the Create Package Definition option is run. The Package Definitions tab remains empty.



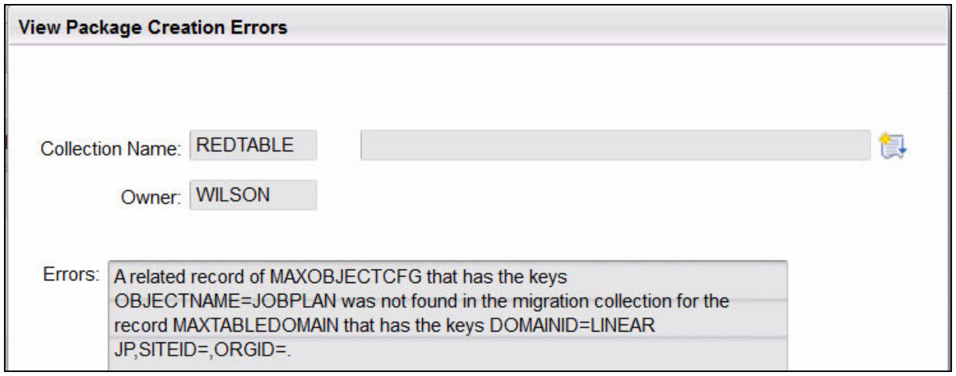
Application	Primary Keys	Object Structure
DOMAINADM	DOMAINID=LINEAR JP	DMMAXDOMA

Details

Application:	DOMAINADM
Object:	MAXDOMAIN
Primary Keys:	DOMAINID=LINEAR JP
Object Structure:	DMMAXDOMA
Create Mode:	MANUAL

Figure 3-14 Table domain with dependency on object

Upon opening the View Package Creation Errors window, the error message indicates that the business object, JOBPLAN, also should be part of the collection to ensure that circular dependency is met. Figure 3-15 shows the error message.


View Package Creation Errors

Collection Name: REDTABLE

Owner: WILSON

Errors: A related record of MAXOBJECTCFG that has the keys OBJECTNAME=JOBPLAN was not found in the migration collection for the record MAXTABLEDOMAIN that has the keys DOMAINID=LINEAR JP,SITEID=,ORGID=.

Figure 3-15 Package definition creation error message

After the JOBPLAN object is added to the collection, the Create Package Definition option is run again. This time, two package definitions are successfully created and displayed in the Package Definitions tab.

Although this example used a domain and object that are supplied with the product, custom objects and domains with circular dependencies should be especially handled correctly so that the appropriate package definitions can be generated.

4

Migration Manager Comparison

This chapter introduces you to a new feature of Migration Manager 7.5, called Comparison. Comparison functionality can be used to accelerate the migration by uncovering differences in system configurations between a source and target product environment. The comparison is run based on the specific package definition that you constructed in Migration Manager. Database connectivity is required from one product environment to the other to run comparison.

This chapter includes the following sections:

- Comparison drivers and benefits
- Comparison reporting

4.1 Comparison drivers and benefits

It is a common practice to prepare and begin migrating configuration changes after a considerable number of changes are made. There might not always be a detailed record of what configurations were created or modified. Under these circumstances, the implementer can use the Comparison functionality to compare the system configurations of local and remote product environments. The identified differences are then collected in a migration collection and are used to prepare subsequent migration packages. The implementer can also use the Comparison functionality as a post-migration verification tool to compare source and target environments and ensure that the system configurations match. With either scenario, the comparison functionality significantly reduces the time that is needed to perform and verify migrations.

When comparisons are conducted, the notion of source versus target environments is not a necessarily a germane consideration. You can use Comparison to view what changed in either of your environments regardless as to which one is source or target. The comparison is said to run between a local system versus a remote system.

Comparison includes the following benefits:

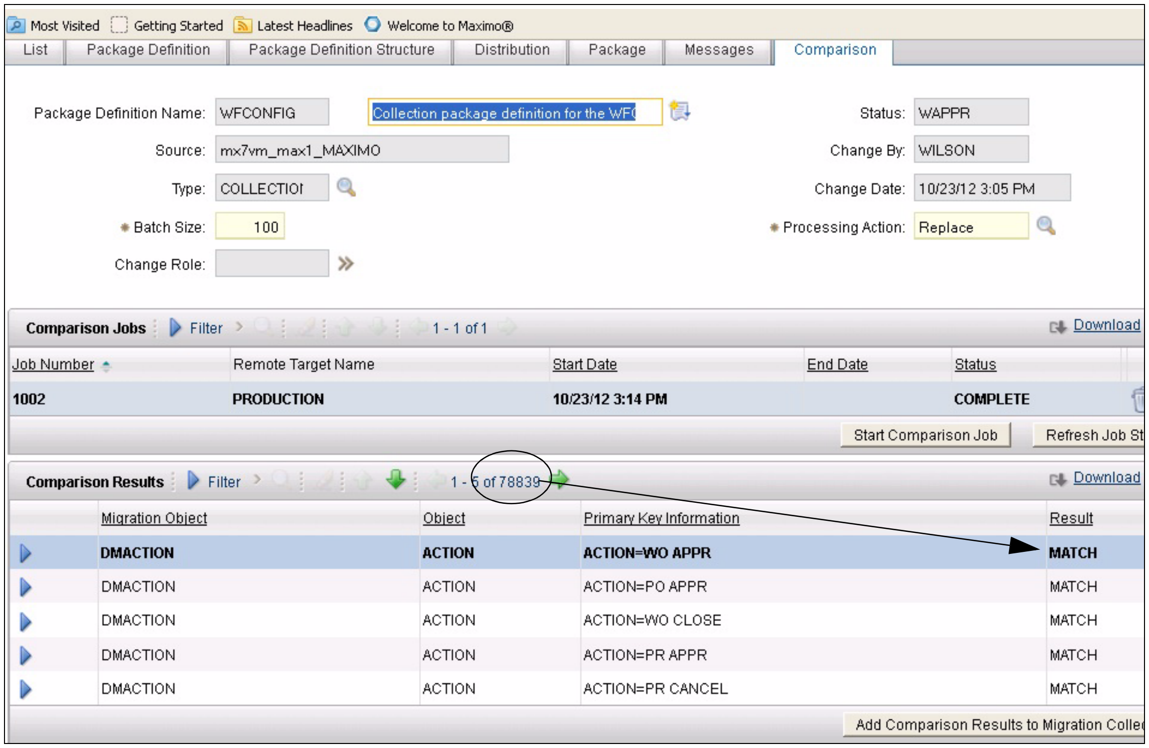
- Traceability
- Avoidance of work duplication
- Analysis of differences between system configurations

For more information about Comparison and its limitations, see 15.9, “Migration Manager comparisons” on page 394.

4.1.1 Traceability

There are instances where individuals who are responsible for creating configuration changes in the development environment might not record their activities or might move to other positions. Therefore, there is no clear record of what changes took place in the development environment.

You can run a Comparison report based on comparison results that are visible in the Migration Manager application. Figure 4-1 shows comparison results that were filtered to show matches between two product environments. In this example, there are 78,839 records that match.



Package Definition Name:	WFCONFIG	Collection package definition for the WF(	
Source:	mx7vm_max1_MAXIMO	Status:	WAPPR
Type:	COLLECTIOI	Change By:	WILSON
Batch Size:	100	Change Date:	10/23/12 3:05 PM
Change Role:		Processing Action:	Replace

Comparison Jobs — 1 - 1 of 1

Job Number	Remote Target Name	Start Date	End Date	Status
1002	PRODUCTION	10/23/12 3:14 PM		COMPLETE

Comparison Results — 1 - 5 of 78839

Migration Object	Object	Primary Key Information	Result
DMACTION	ACTION	ACTION=WO APPR	MATCH
DMACTION	ACTION	ACTION=PO APPR	MATCH
DMACTION	ACTION	ACTION=WO CLOSE	MATCH
DMACTION	ACTION	ACTION=PR APPR	MATCH
DMACTION	ACTION	ACTION=PR CANCEL	MATCH

Figure 4-1 Sample comparison results

A migration package definition that is called WFCONFIG was used to perform the comparison. This particular migration package contains system configurations that are commonly used in implementing workflow processes, such as Actions, Roles, Communication Templates, and the Workflow process.

Comparison functionality can uncover matches and differences, which reveals system configuration activities in source and target product environments.

4.1.2 Avoidance of work duplication and over writing records

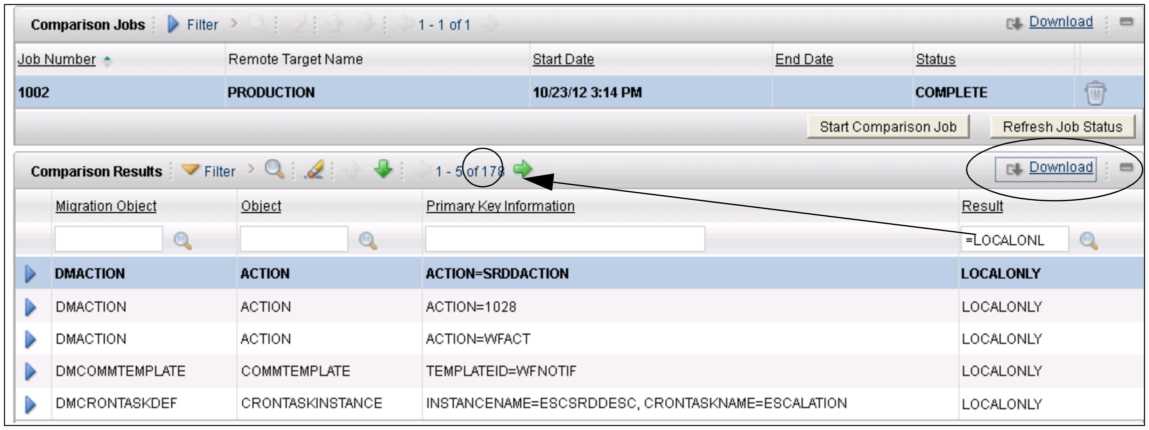
There are instances where a group of developers might create configurations directly in production environments. Creating these configurations does not lead to best practices, yet it does occur often enough to warrant a concern or even a business pain point.

Comparison is well-suited to resolving this issue. By discovering the differences that pertain to specific migration packages between the local and the remote environment, implementers can make more informed decisions regarding an implementation effort and avoid the possibility of overwriting another's configuration changes.

4.1.3 Analysis of differences

You can run an Ad Hoc report from the results of your migration comparison tool and save the report as spreadsheet or document. This report can help with filtering and reviewing the comparison results.

In the example that is shown in Figure 4-2, the comparison results were filtered to view those configurations that exist exclusively in the local environment. The filter reveals 178 configurations.



Job Number	Remote Target Name	Start Date	End Date	Status
1002	PRODUCTION	10/23/12 3:14 PM		COMPLETE

Migration Object	Object	Primary Key Information	Result
			=LOCALONL
DMACTION	ACTION	ACTION=SRDDACTION	LOCALONLY
DMACTION	ACTION	ACTION=1028	LOCALONLY
DMACTION	ACTION	ACTION=WFACT	LOCALONLY
DMCOMMTEMPLATE	COMMTEMPLATE	TEMPLATEID=WFNOTIF	LOCALONLY
DMCRONTASKDEF	CRONTASKINSTANCE	INSTANCENAME=ESCSRDDESC, CRONTASKNAME=ESCALATION	LOCALONLY

Figure 4-2 View of system configurations that are present only in the local environment

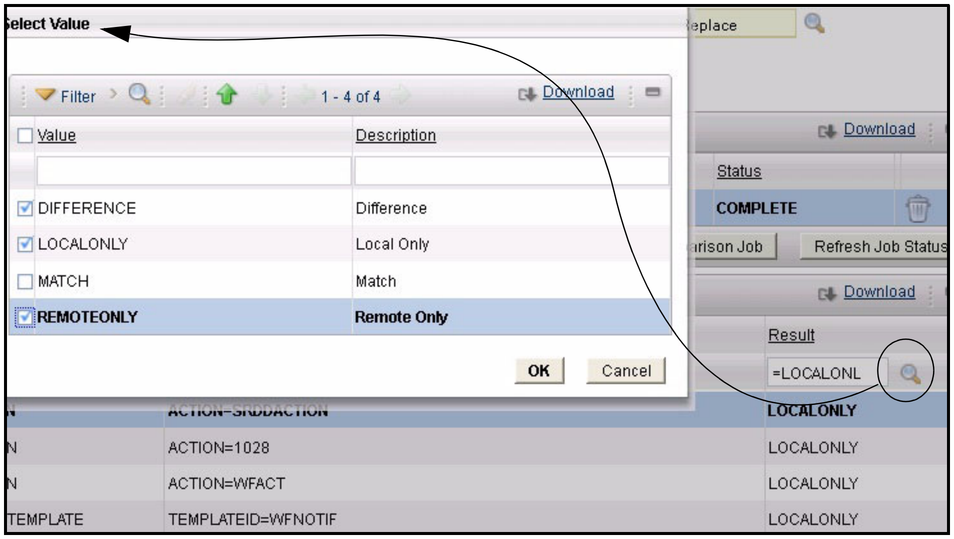
Comparison results

Comparison results can list one of the following possible outcomes, as shown in Table 4-1:

- Differences
- Local Only
- Match
- Remote Only (see Figure 4-3 on page 80)

Table 4-1 Possible Comparison result outcomes

Result type	Description
Differences	Choosing this filter option enables the implementer to view only the differences between the two databases. Two records of a system configuration are deemed to be different when they are identified by the same primary key values but other values are different. Ideally, at certain points during the implementation cycle, you should expect to find no differences between the two participating environments. No differences might not always be achievable based on ongoing development and maintenance activity in the product environments.
Local Only	Choosing this filter option enables the implementer to view only those system configurations that are defined in the local environment from where Comparison was initiated. In this context, the local environment also might be the source environment. If the comparison functionality is used as a verification tool, it can be run directly from a production environment against a development environment. Local Only outcome indicates a configuration in product (local) that is not in development (remote).
Match	Choosing this filter option enables the implementer to view those system configurations that are the same between the local and remote environments.
Remote Only	Choosing this filter option enables the implementer to view only those system configurations that are defined in remote environment to which the comparison was pointed, as shown in Figure 4-3 on page 80.

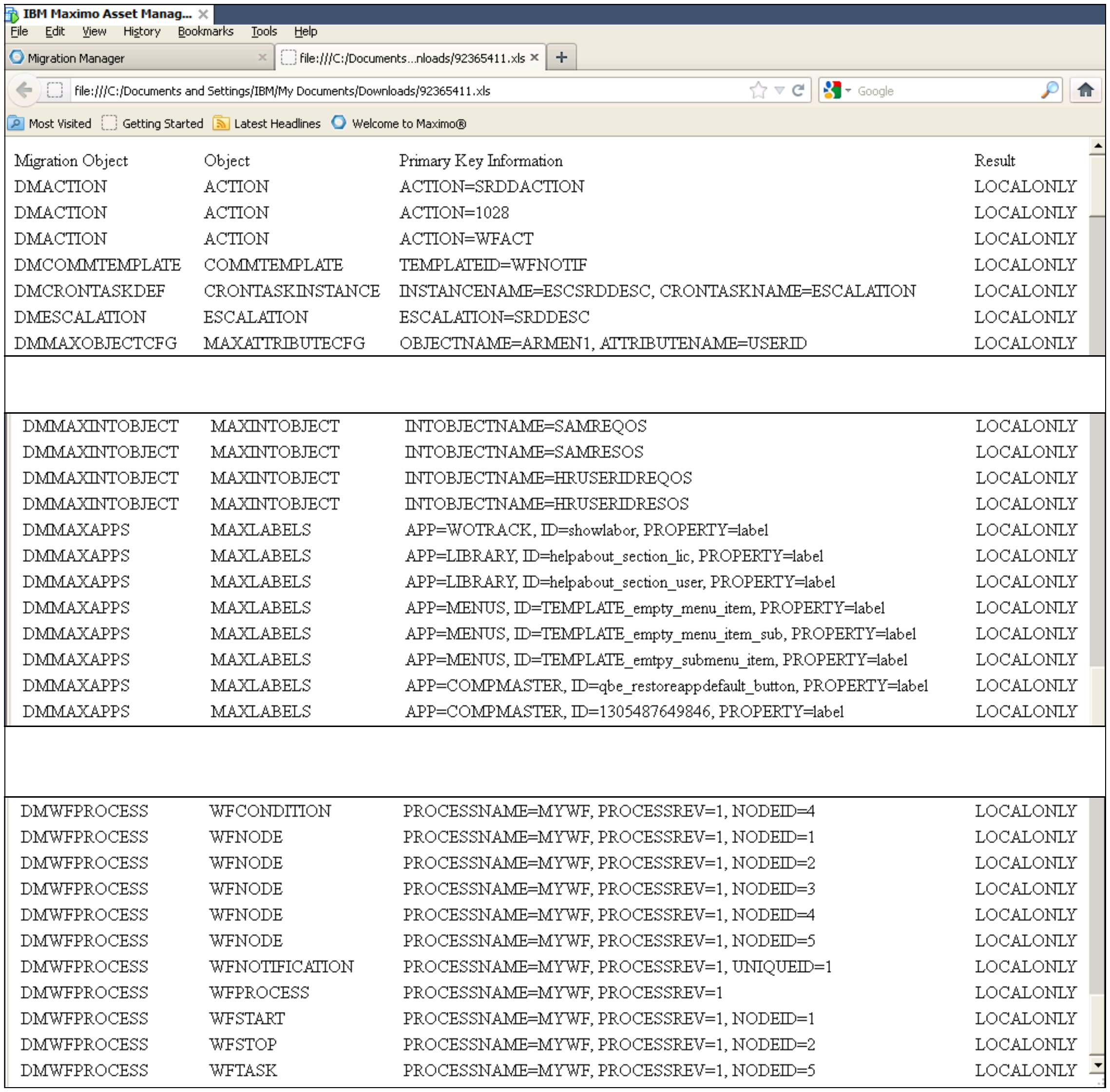


	Value	Description
✓	DIFFERENCE	Difference
✓	LOCALONLY	Local Only
	MATCH	Match
✓	REMOTEONLY	Remote Only

N	ACTION=1028	LOCALONLY
N	ACTION=WFACT	LOCALONLY
TEMPLATE	TEMPLATEID=WFNOTIF	LOCALONLY

Figure 4-3 Using the Remote Only option

Figure 4-4 shows an example Ad Hoc report that an implementer generated to display the system configurations that are present only in the local environment, which might be source or target.


IBM Maximo Asset Manag...
File Edit View History Bookmarks Tools Help
Migration Manager
file:///C:/Documents...nloads/92365411.xls
file:///C:/Documents and Settings/IBM/My Documents/Downloads/92365411.xls
Google
Most Visited
Getting Started
Latest Headlines
Welcome to Maximo®

Migration Object	Object	Primary Key Information	Result
DMACTION	ACTION	ACTION=SRDDACTION	LOCALONLY
DMACTION	ACTION	ACTION=1028	LOCALONLY
DMACTION	ACTION	ACTION=WFACT	LOCALONLY
DMCOMMTEMPLATE	COMMTEMPLATE	TEMPLATEID=WFNOTIF	LOCALONLY
DMCRONTASKDEF	CRONTASKINSTANCE	INSTANCENAME=ESCSRDDESC, CRONTASKNAME=ESCALATION	LOCALONLY
DMESCALATION	ESCALATION	ESCALATION=SRDDESC	LOCALONLY
DMMAXOBJECTCFG	MAXATTRIBUTECFG	OBJECTNAME=ARMEN1, ATTRIBUTENAME=USERID	LOCALONLY
DMMAXINTOBJECT	MAXINTOBJECT	INTOBJECTNAME=SAMREQOS	LOCALONLY
DMMAXINTOBJECT	MAXINTOBJECT	INTOBJECTNAME=SAMRESOS	LOCALONLY
DMMAXINTOBJECT	MAXINTOBJECT	INTOBJECTNAME=HRUSERIDREQOS	LOCALONLY
DMMAXINTOBJECT	MAXINTOBJECT	INTOBJECTNAME=HRUSERIDRESOS	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=WOTRACK, ID=showlabor, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=LIBRARY, ID=helpabout_section_lic, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=LIBRARY, ID=helpabout_section_user, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=MENUS, ID=TEMPLATE_empty_menu_item, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=MENUS, ID=TEMPLATE_empty_menu_item_sub, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=MENUS, ID=TEMPLATE_emtpy_submenu_item, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=COMPMASTER, ID=qbe_restoreappdefault_button, PROPERTY=label	LOCALONLY
DMMAXAPPS	MAXLABELS	APP=COMPMASTER, ID=1305487649846, PROPERTY=label	LOCALONLY
DMWFPROCESS	WFCONDITION	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=4	LOCALONLY
DMWFPROCESS	WFNODE	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=1	LOCALONLY
DMWFPROCESS	WFNODE	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=2	LOCALONLY
DMWFPROCESS	WFNODE	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=3	LOCALONLY
DMWFPROCESS	WFNODE	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=4	LOCALONLY
DMWFPROCESS	WFNODE	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=5	LOCALONLY
DMWFPROCESS	WFNOTIFICATION	PROCESSNAME=MYWF, PROCESSREV=1, UNIQUEID=1	LOCALONLY
DMWFPROCESS	WFPROCESS	PROCESSNAME=MYWF, PROCESSREV=1	LOCALONLY
DMWFPROCESS	WFSTART	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=1	LOCALONLY
DMWFPROCESS	WFSTOP	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=2	LOCALONLY
DMWFPROCESS	WFTASK	PROCESSNAME=MYWF, PROCESSREV=1, NODEID=5	LOCALONLY

Figure 4-4 Example report

4.1.4 Using Migration Comparison to ensure identical system configurations

Often times, the requirement is to ensure that two or more development environments feature the exact same configuration changes. This requirement is a key measure of whether a production environment reflects all development that was completed to date.

A similar requirement might arise with distributed development environments where a baseline must be established to ensure all development teams start with the same system configurations.

By running Comparison and filtering the results for differences, zero different entries should exist, which is not always achievable or the appropriate goal.

There might be times that certain differences must be explained and preserved. For example, certain users and security groups might be defined in the development environment that would never exist in the production environment. The Comparison can help identify and validate such differences. Another example is when certain authorizations (signature options) that are granted to certain users might be different in the development (local) environment than in the production (remote) environment.

Such differences must be explained as valid and preserved, not necessarily made equal in the local and remote system.

4.1.5 Migration Comparison is superior to running an SQL compare

You can run specific SQL commands in the form of database scripts to find records that are present in a specific database as compared with another database. Often, this ability requires the creation of database-level links between the two participating databases and must be configured by a database administrator. However, this approach includes the following significant disadvantages when compared with Migration Manager’s comparison functionality:

- No adherence to the structure of business objects, including parent-child relationships.
- Cannot uncover multiple outcomes, such as Difference, Match, Local Only, and Remote Only.
- No ability to quickly filter and generate formatted reports.

4.2 Comparison reporting

Comparison results are stored in the database and displayed in the Migration Manager user interface. To enable the generation of Ad Hoc reports from the persisted data, a new reporting object structure named REP_DMCOMPRESULT is now delivered with the product. This object structure encapsulates the comparison results tables, which offer implementers a quick path to generating quality reports.

Best practice: For more information about Ad Hoc report functionality, usage and best practices, see the Ad Hoc report reference materials that are available at this website:

https://www.ibm.com/developerworks/wikis/display/tivolispa/Ad+Hoc+%28QBR%29+Reference+Materials

4.2.1 Creating and viewing a comparison results report by using QBR

In this scenario, a new person record in the source (development) environment to simulate a new system configuration. Next, the comparison function in Migration Manager is run to reveal what changed between the source and the target environment.

The following steps are used in this process:

1. Create a person record in the source environment.
2. Include the PERSON object in migration manager.
3. Run a comparison to see what changed between the source and the target environment.
4. Run an Ad Hoc report to view the differences and refine the report to view what changed in the local-only (source) environment.

4.2.2 Creating a person record

Complete the following steps to create a person record:

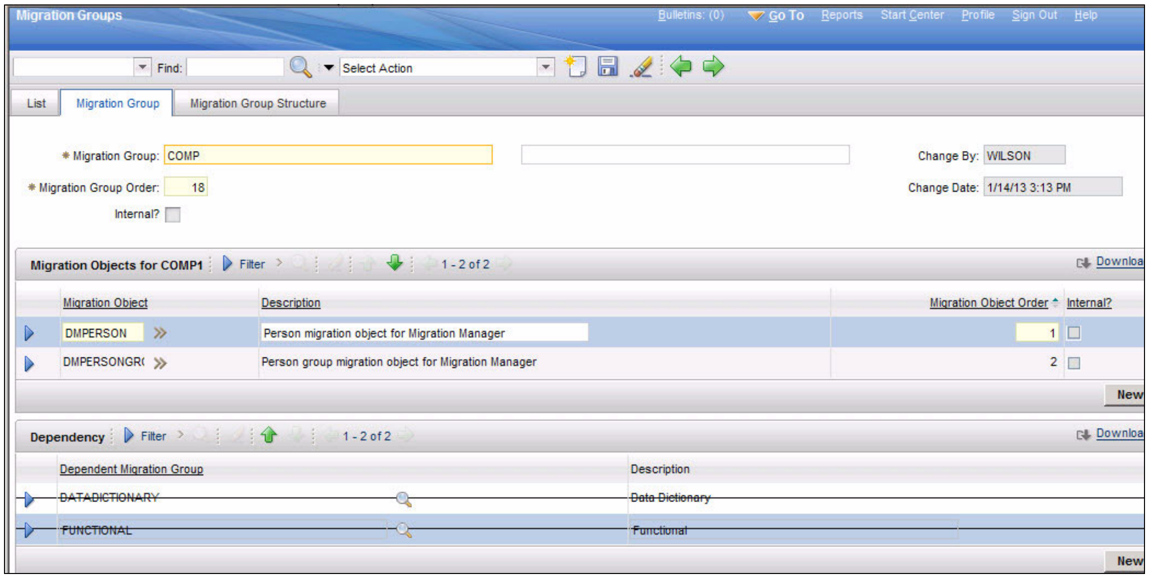
1. Click **Administration** → **Resources** → **People** to open the People application.
2. Create a person record that uses your name. You must specify a name only in the **Person** field. Save the record.

You now have a record that is specific to the source environment that does not exist in the target environment.

4.2.3 Creating a migration group

Complete the following steps to create a migration group:

1. Open the Migration Groups application.
2. Retrieve the Resources group and duplicate it. Delete both Dependencies that are shown in the Dependency table window. These dependencies also exist in the remote or target environment and there is no need to include them in the package definition, as shown in Figure 4-5.



Migration Group:	COMP
Migration Group Order:	18
Internal?	☐
Change By:	WILSON
Change Date:	1/14/13 3:13 PM

Migration Objects for COMP1 — 1 - 2 of 2

Migration Object	Description	Migration Object Order	Internal?
DMPERSON	Person migration object for Migration Manager	1	☐
DMPERSONGR(	Person group migration object for Migration Manager	2	☐

Dependency — 1 - 2 of 2

Dependent Migration Group	Description
~~DATADICTIONARY~~	~~Data Dictionary~~
~~FUNCTIONAL~~	~~Functional~~

Figure 4-5 Duplicating the Resources migration group

4.2.4 Creating a migration package definition

Complete the following steps to create a migration package definition:

1. Open the Migration Manager application.
2. Create a record.
3. Click **New Row** in the Migration Group table window and select the migration group that you created in 4.2.3, “Creating a migration group” on page 84.

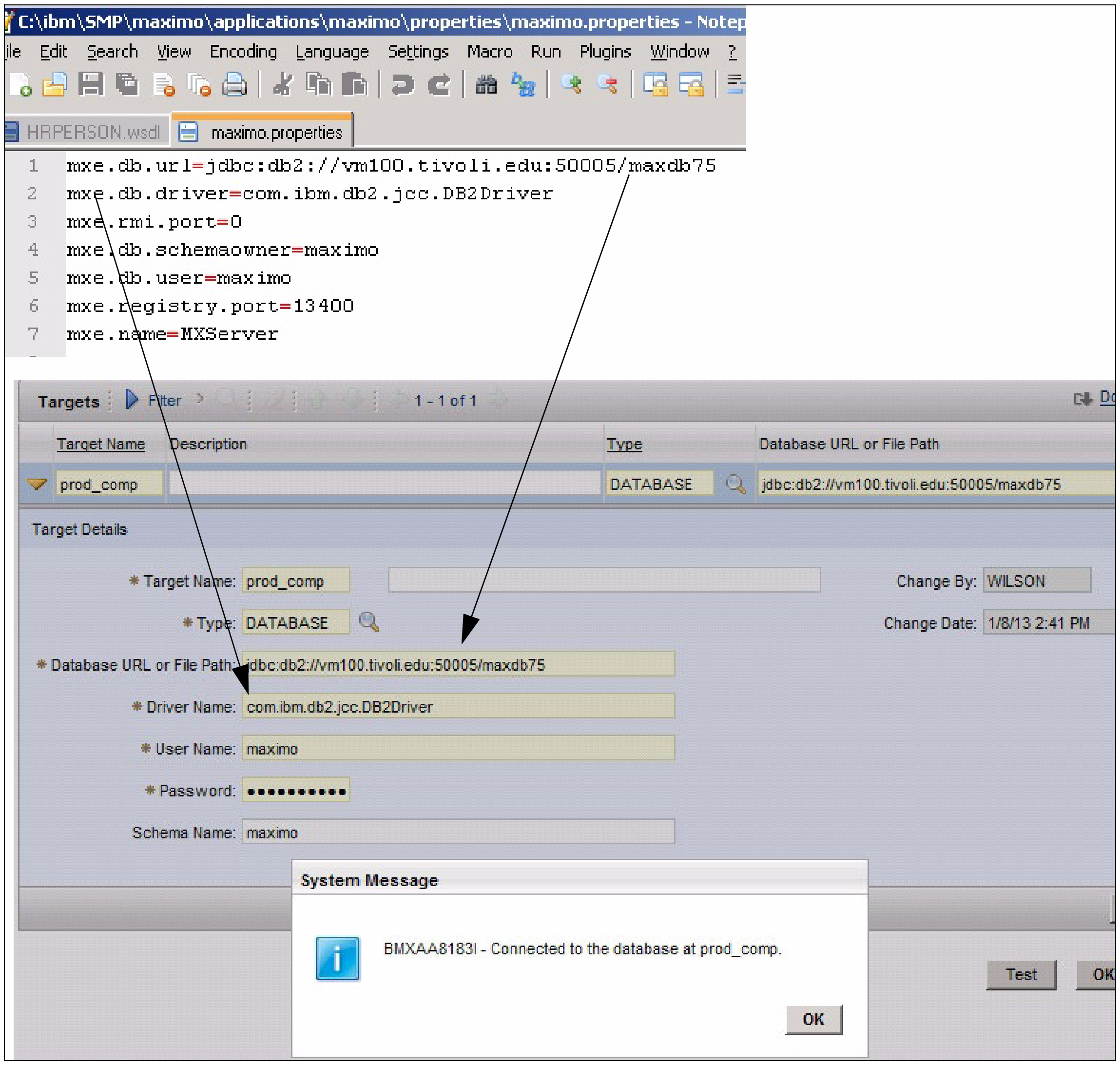
4.2.5 Defining database connectivity to the remote environment

Before you start a comparison job, you must create a database connection between both environments. Complete the following steps to create the database connection:

1. Click the **Manage Targets** icon that is in the toolbar (🎯).
2. Click **New Row**. Complete the following steps:
   a. Enter a target name: `Prod1`
   b. Enter a description: `Only for comparison.`
   c. To take advantage of the Comparison feature, which is new to Migration Manager 7.5, select **DATABASE** in the Type field instead of File for accessing the target.

Important: There cannot be any comparison if you point to a folder on the file system. The database is a Windows service, and is set to start automatically during system start. You do not need to start the application server now. The database instance is running.

You obtain the database connectivity information from the `maximo.properties` file that is in the `root_maximo_installation/applications/maximo/properties` directory of the target or remote environment (not the source environment). In this example, the `maximo.properties` file features the following name and location: `C:\ibm\SMP\maximo2\applications\maximo\properties\maximo.properties`, as shown in Figure 4-6. Alternately, the system administrator can provide the required details.



C:\ibm\SMP\maximo\applications\maximo\properties\maximo.properties - Notep

```
mxe.db.url=jdbc:db2://vm100.tivoli.edu:50005/maxdb75
mxe.db.driver=com.ibm.db2.jcc.DB2Driver
mxe.rmi.port=0
mxe.db.schemaowner=maximo
mxe.db.user=maximo
mxe.registry.port=13400
mxe.name=MXServer
```

Targets — 1 - 1 of 1

Target Name	Description	Type	Database URL or File Path
prod_comp		DATABASE	jdbc:db2://vm100.tivoli.edu:50005/maxdb75

Target Details

* Target Name:	prod_comp
* Type:	DATABASE
* Database URL or File Path:	jdbc:db2://vm100.tivoli.edu:50005/maxdb75
* Driver Name:	com.ibm.db2.jcc.DB2Driver
* User Name:	maximo
* Password:	●●●●●●●●●●●
Schema Name:	maximo
Change By:	WILSON
Change Date:	1/8/13 2:41 PM

System Message

BMXAA8183I - Connected to the database at prod_comp.

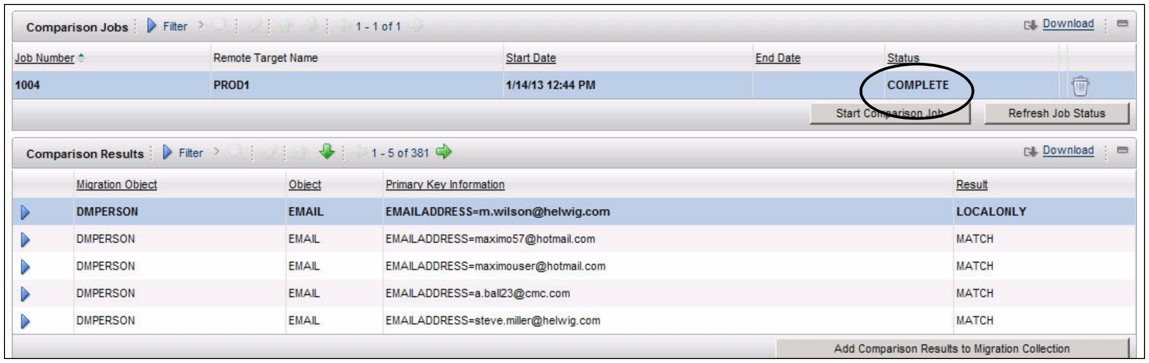
Figure 4-6 Name and location of the maximo.properties file

   d. For the database web address or file path (in this example), enter `jdbc:db2://vm100.tivoli.edu:50005/maxdb75`
   e. Enter a driver name: `com.ibm.db2.jcc.DB2Driver`
   f. Enter a user name: `maximo`
   g. Enter the password: `object00`
   h. Enter a schema name: `maximo`
3. Click **Test** and ensure that you receive the System Message, which indicates a successful connection.

4.2.6 Running the comparison

Complete the following steps to run the comparison:

1. Click the Comparison tab.
2. Click **Start Comparison Job**.
3. Click **Start Job**. This process might run for a few minutes.
4. Periodically, click **Refresh Job Status** and notice the Comparison Results count.
5. Wait until the status changes from INPROGRESS to COMPLETE, as shown in Figure 4-7.

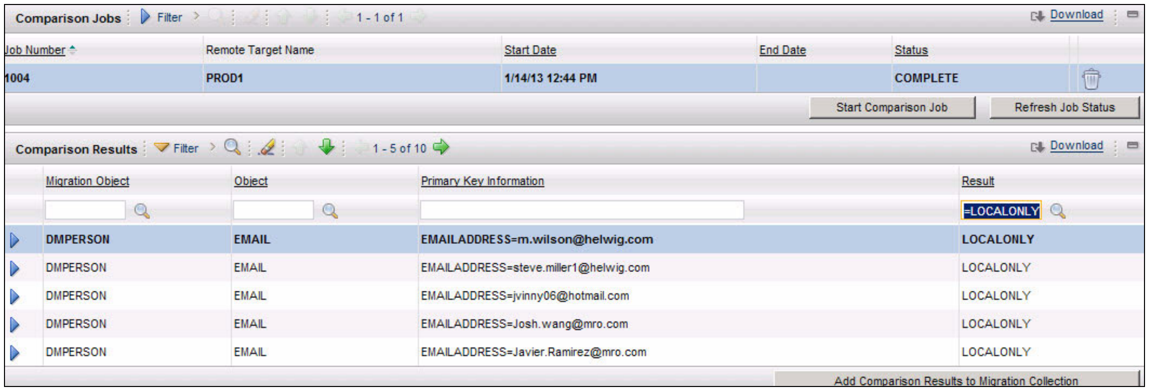


Job Number	Remote Target Name	Start Date	End Date	Status
1004	PROD1	1/14/13 12:44 PM		COMPLETE

Migration Object	Object	Primary Key Information	Result
DMPERSON	EMAIL	EMAILADDRESS=m.wilson@helwig.com	LOCALONLY
DMPERSON	EMAIL	EMAILADDRESS=maximo57@hotmail.com	MATCH
DMPERSON	EMAIL	EMAILADDRESS=maximouser@hotmail.com	MATCH
DMPERSON	EMAIL	EMAILADDRESS=a.ball23@cmc.com	MATCH
DMPERSON	EMAIL	EMAILADDRESS=steve.miller@helwig.com	MATCH

Figure 4-7 Status COMPLETE

6. Open the filter, change the Results criteria, and run another comparison report that uses LOCALONLY in the Results field. Press Enter to populate the Results criteria. In the example that is shown in Figure 4-8, there are 10 configuration changes that are unique to the local or source environment.


Comparison Jobs | Filter | 1 - 1 of 1 | Download

Job Number	Remote Target Name	Start Date	End Date	Status
1004	PROD1	1/14/13 12:44 PM		COMPLETE

Start Comparison Job | Refresh Job Status

Comparison Results | Filter | 1 - 5 of 10 | Download

Migration Object	Object	Primary Key Information	Result
			=LOCALONLY
DMPERSON	EMAIL	EMAILADDRESS=m.wilson@helwig.com	LOCALONLY
DMPERSON	EMAIL	EMAILADDRESS=steve.miller1@helwig.com	LOCALONLY
DMPERSON	EMAIL	EMAILADDRESS=jvinny06@hotmail.com	LOCALONLY
DMPERSON	EMAIL	EMAILADDRESS=Josh.wang@mro.com	LOCALONLY
DMPERSON	EMAIL	EMAILADDRESS=Javier.Ramirez@mro.com	LOCALONLY

Add Comparison Results to Migration Collection

Figure 4-8 Configuration changes

4.2.7 Generating a comparison report

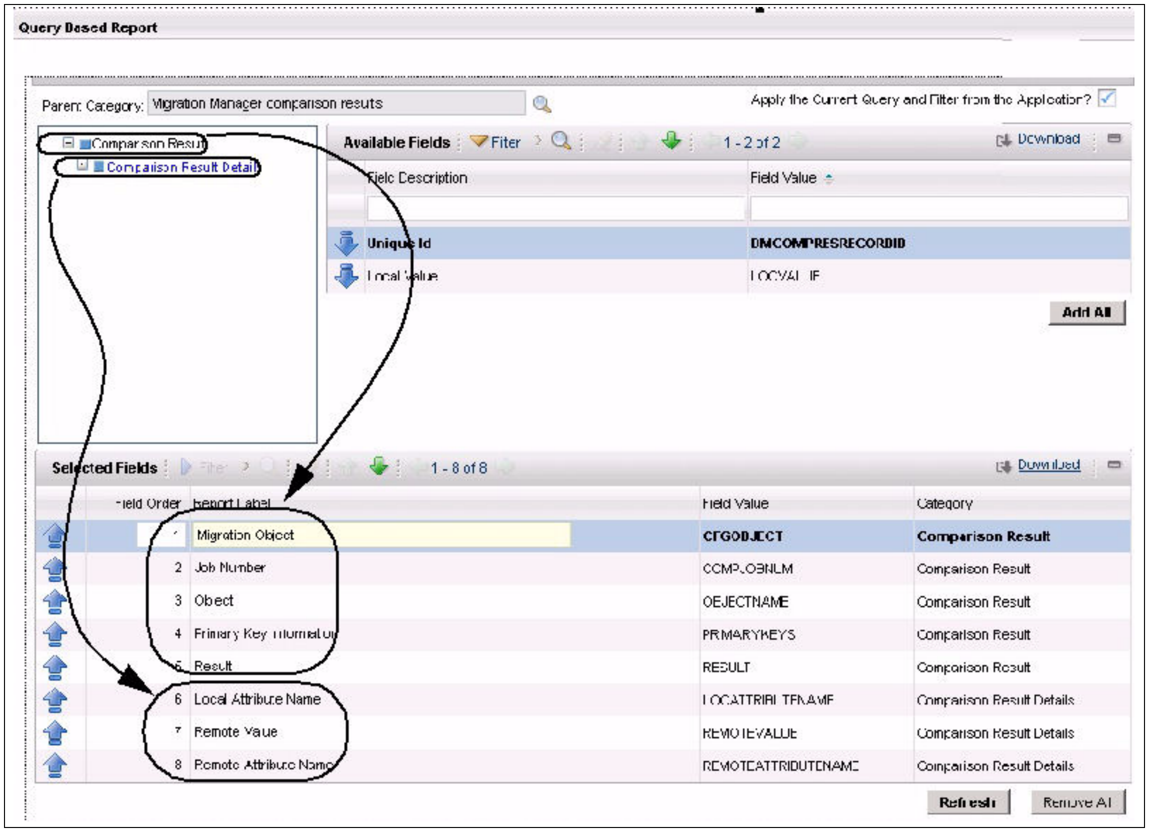
Complete the following steps to create an Ad Hoc report for a typical migration manager comparison results. The REP_DMCOMPRESULT object structure is used to generate this report:

1. From the Migration Manager application, click the **Create Report** icon in the toolbar.
2. In the Query Based report window, select **Summary Report** as the style.
3. Enter an appropriate report title and select **Save Report** to save the report before it is run for the first time.

4. Click the **Select** tab and highlight the **Comparison Results** entry. Complete the following steps:

   a. As shown in Figure 4-9, add the following fields to your report by clicking the down arrow: **Migration Object**, **Job Number**, **Object**, **Primary Key Information**, and **Result** (next page) columns.

   b. Continuing from the Select tab, highlight the **Comparison Results Details entry** and add the following fields from the Available Fields table window to the Selected Fields table window by using the down arrow: **Local Attribute Name**, **Remote Value**, and **Remote Attribute Name**.



Query Based Report

Parent Category: Migration Manager comparison results

Apply the Current Query and Filter from the Application? ✓

- Comparison Result
  - Comparison Result Detail

Available Fields — Filter — 1 - 2 of 2 — Download

	Field Description	Field Value
	Unique Id	DMCOMPRESRECORDID
	Local Value	LOCALVALUE

Add All

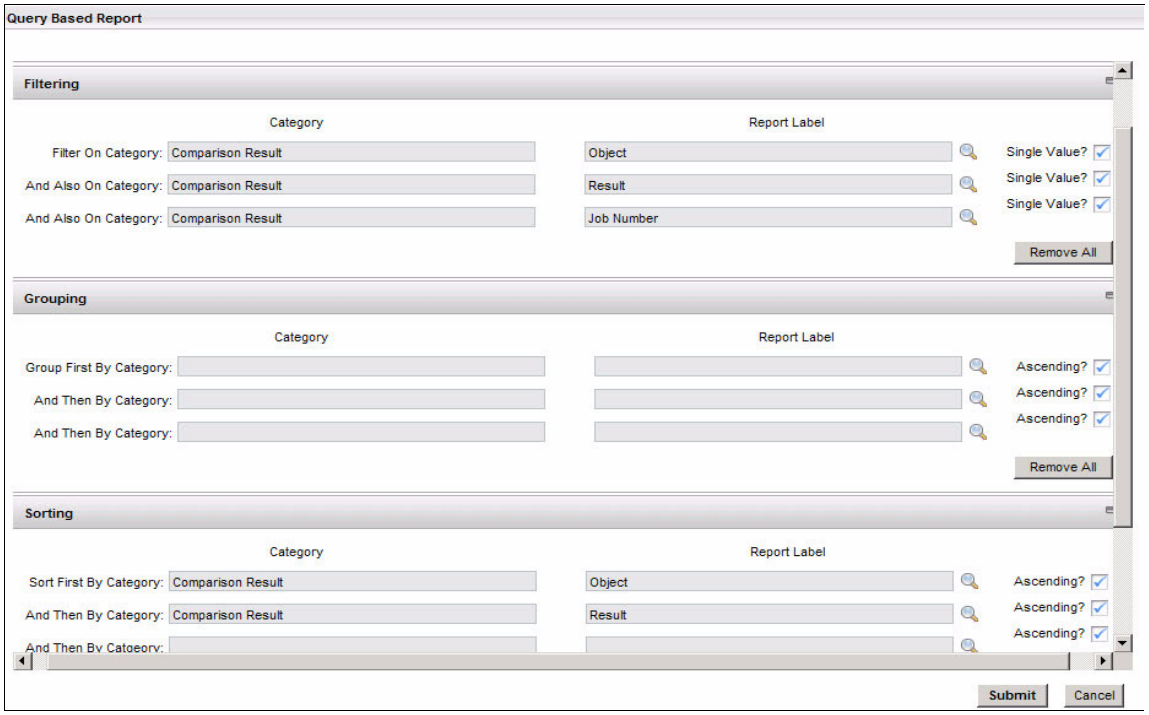
Selected Fields — Filter — 1 - 8 of 8 — Download

	Field Order	Report Label	Field Value	Category
	1	Migration Object	CFGOBJECT	Comparison Result
	2	Job Number	COMPJOBNUM	Comparison Result
	3	Object	OBJECTNAME	Comparison Result
	4	Primary Key Information	PRIMARYKEYS	Comparison Result
	5	Result	RESULT	Comparison Result
	6	Local Attribute Name	LOCATTRIBUTENAME	Comparison Result Details
	7	Remote Value	REMOTEVALUE	Comparison Result Details
	8	Remote Attribute Name	REMOTEATTRIBUTENAME	Comparison Result Details

Refresh — Remove All

Figure 4-9 Adding fields

5. Click the **Format** tab of the Query Based report window. Complete the following steps:
    a. From the lookup icon to the right of the Filter On Category, select **Object**.
    b. From the lookup icon to the right of the And Also on Category, select **Result**.
    c. From the lookup icon to the right of the And Also on Category, select **Job Number**.
    d. In the Sorting section of the same Format tab, select the lookup icon to the right of the Sort First By Category field and select **Object** from the DMCOMPRESULT entry.
    e. In the Sorting section of the same Format tab, select the lookup icon to the right of the And Then By Category field and select **Result** from the DMCOMPRESULT entry, as shown in Figure 4-10 on page 90.


Query Based Report

Filtering

	Category	Report Label	
Filter On Category:	Comparison Result	Object	Single Value? ✓
And Also On Category:	Comparison Result	Result	Single Value? ✓
And Also On Category:	Comparison Result	Job Number	Single Value? ✓

Grouping

	Category	Report Label	
Group First By Category:			Ascending? ✓
And Then By Category:			Ascending? ✓
And Then By Category:			Ascending? ✓

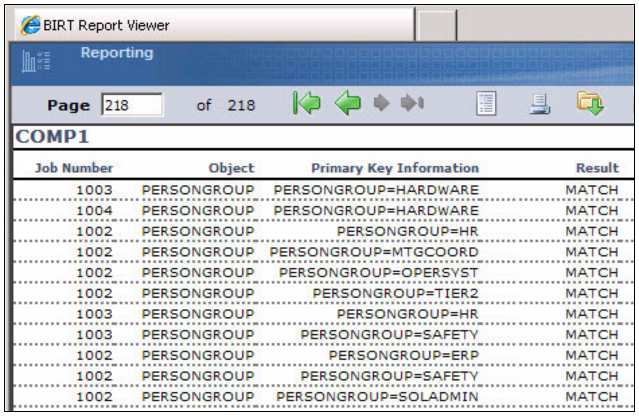
Sorting

	Category	Report Label	
Sort First By Category:	Comparison Result	Object	Ascending? ✓
And Then By Category:	Comparison Result	Result	Ascending? ✓
And Then By Category:			Ascending? ✓

Figure 4-10 Select Result from the DMCOMPRESULT MBO

6. Click the **Submit** tab of the Query Based Report window.

   The Ad Hoc report runs and a new window (or tab) opens, which shows the results of the report, as shown in Figure 4-11.



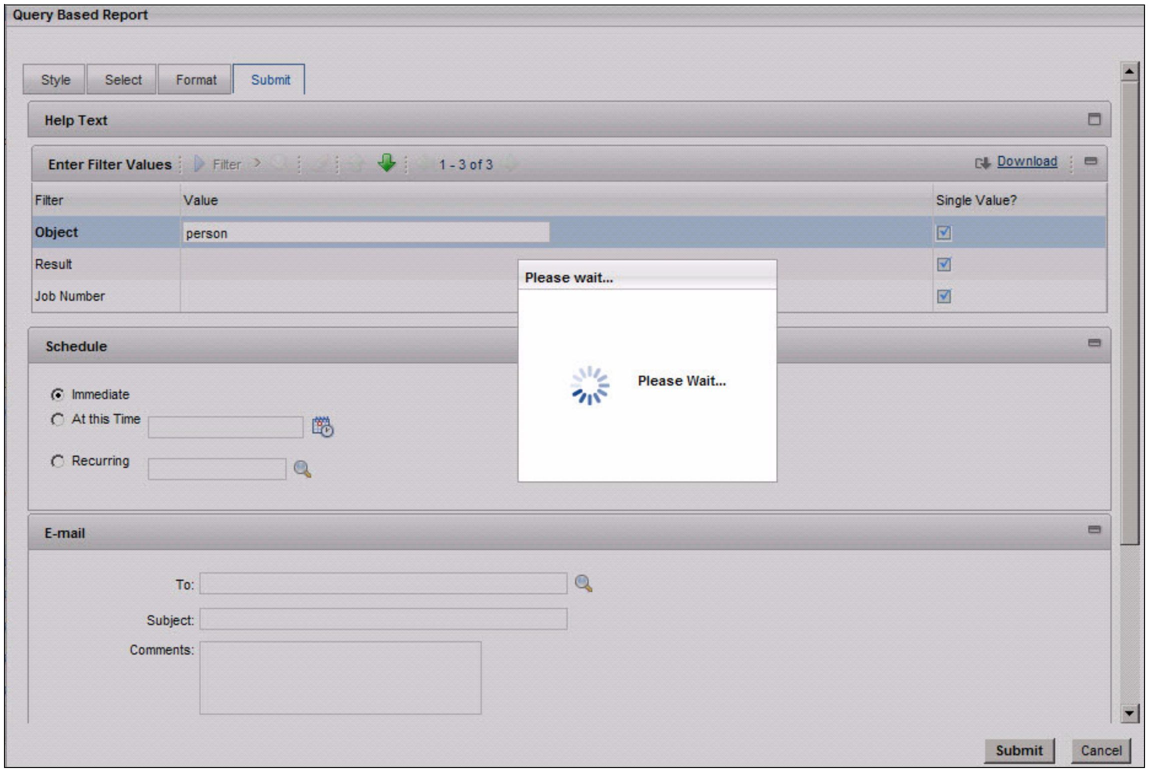
COMP1

Job Number	Object	Primary Key Information	Result
1003	PERSONGROUP	PERSONGROUP=HARDWARE	MATCH
1004	PERSONGROUP	PERSONGROUP=HARDWARE	MATCH
1002	PERSONGROUP	PERSONGROUP=HR	MATCH
1002	PERSONGROUP	PERSONGROUP=MTGCOORD	MATCH
1002	PERSONGROUP	PERSONGROUP=OPERSYST	MATCH
1002	PERSONGROUP	PERSONGROUP=TIER2	MATCH
1003	PERSONGROUP	PERSONGROUP=HR	MATCH
1003	PERSONGROUP	PERSONGROUP=SAFETY	MATCH
1002	PERSONGROUP	PERSONGROUP=ERP	MATCH
1002	PERSONGROUP	PERSONGROUP=SAFETY	MATCH
1002	PERSONGROUP	PERSONGROUP=SOLADMIN	MATCH

Figure 4-11 Results of the Ad Hoc report

You can now further define your query to view changes that were made to the PERSON object and its related tables.

7. In the Submit tab, specify PERSON as the value for Object filter and click **Submit**, as shown in Figure 4-12.


Query Based Report

Filter	Value	Single Value?
Object	person	✓
Result		✓
Job Number		✓

Please wait...

Please Wait...

Schedule

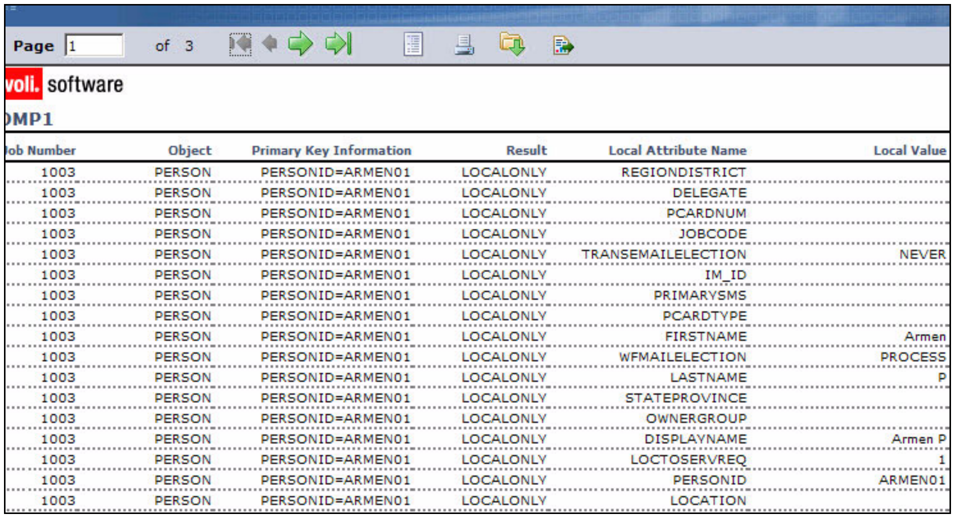
[✓ Immediate]
At this Time
Recurring

E-mail

To:
Subject:
Comments:

Figure 4-12 Clicking Submit

As shown in Figure 4-13, you receive a more granular report that shows the changes that you made only to the PERSON record.



Page 1 of 3

voli. software

MP1

ob Number	Object	Primary Key Information	Result	Local Attribute Name	Local Value
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	REGIONDISTRICT	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	DELEGATE	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	PCARDNUM	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	JOBCODE	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	TRANSEMAILELECTION	NEVER
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	IM_ID	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	PRIMARYSMS	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	PCARDTYPE	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	FIRSTNAME	Armen
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	WFMAILELECTION	PROCESS
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	LASTNAME	P
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	STATEPROVINCE	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	OWNERGROUP	
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	DISPLAYNAME	Armen P
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	LOCTOSERVREQ	1
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	PERSONID	ARMEN01
1003	PERSON	PERSONID=ARMEN01	LOCALONLY	LOCATION	

Figure 4-13 Report showing changes to the PERSON record only

5

Migrating the data dictionary

The data dictionary contains the building blocks for the environment. It consists of various information, including database objects, attributes, and their relationships. The data dictionary also includes user information, such as organization, site, and currency. Examples of the uses of data dictionary migration include a new object and its attributes, relationships, and domains to use with other migrated data to build a custom application, or the addition of a new international site with its default language and currency.

This chapter includes the following sections:

- Requirements
- Solution A: Snapshot
- Solution B: Collection
- Deployment
- Deployment considerations

5.1 Requirements

The data dictionary objects are defined in a development environment and then migrated to a production environment. For this scenario, we use the pre-formatted Data Dictionary migration package. Not all of the migration objects in the package are covered in this scenario; only the major objects are described.

A good practice in any new development environment is to set up Change migration packages. The Change package can be used to migrate those changes that are made since its activation. Even if they are not used to migrate data, Change packages can be used to track environmental changes for those objects that are supported by the Migration Manager.

Change packages were set up in this scenario before any new content creation to facilitate an accurate list of those configurations to migrate.

5.2 Solution A: Snapshot

Multiple packages are created to facilitate the migration of two sets of configurations.

The first set of configurations consists of the addition of the typical components that are promoted when a new site is added to an existing organization. These objects are migrated in the following order:

1. Currency
2. Sets
3. Site

The second set of configurations consists of configurations that often are migrated from the data dictionary to support adding a tab with a table to an existing application. Also added are the conditions to filter the data based on existing values for particular records. These objects are migrated in following order:

1. Lookup map
2. Domains
3. Objects
4. Relationships
5. Conditional expressions
6. Domains

Domains are included twice in the lists. Because of the complexity of crossover and table domains, extra steps are needed to migrate them correctly. For more information, see 5.4, “Deployment” on page 118.

Because the data dictionary package moves information that is needed by other packages, no external object dependencies must be satisfied as part of the migration. However, other packages are dependent on configurations that are moved as part of the data dictionary migration. Proper planning must be taken to ensure that all migrations occur in the correct order.

5.2.1 Configuration applications

The Database Configuration application manages the following configuration items:

- Objects
- Attributes
- Indexes
- Views
- tables
- Columns
- Autokeys
- Relationships
- Services
- Lookup maps
- Messages
- Sequences
- General Ledger (GL) account configuration

The application can be accessed from the Start Center by selecting **Go To** → **System Configuration** → **Platform Configuration** → **Database Configuration**.

The Domains application manages the domains. These domains can be associated with attributes in the Database Configuration. The domains can be accessed from the Start Center by selecting **Go To** → **System Configuration** → **Platform Configuration** → **Domains**.

The Sets application manages item and company sets that are associated with organizations. The sets can be accessed from the Start Center by selecting **Go To** → **Administration** → **Sets**.

The Currency Codes application manages currency codes that are associated with organizations. The currency codes can be accessed from the Start Center by selecting **Go To** → **Financial** → **Currency Codes**.

The Organizations application manages organizations, sites, and addresses, which can be accessed from the Start Center by selecting **Go To** → **Administration** → **Organizations**.

The Conditional Expression Manager application manages conditional expressions. The conditional expressions can be used with several other configuration objects. The conditional expressions can be accessed from the Start Center by selecting **Go To** → **Administration** → **Conditional Expression Manager**.

The Object Structures application manages object structures that are used by the Integration Framework to import data and by the Query-Based Report functionality to join tables for ad hoc reporting. The object structures can be accessed from the Start Center by selecting **Go To** → **Integration** → **Object Structures**.

5.2.2 Object structures

There are 15 predefined object structures that are part of the Data Dictionary migration package.

The DMMAXSERVICE object structure, as shown in Figure 5-1, supports services migration.

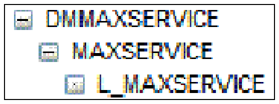


Figure 5-1 DMMAXSERVICE

The DMLANGUAGE object structure, as shown in Figure 5-2, supports language migration.

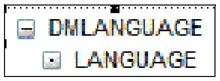


Figure 5-2 DMLANGUAGE

The DMMAXMESSAGES object structure, as shown in Figure 5-3, supports messages migration.

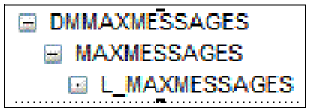


Figure 5-3 DMMAXMESSAGES

The DMMAXLOOKUPMAP object structure, as shown in Figure 5-4, supports lookup migration.

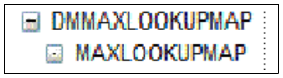

DMMAXLOOKUPMAP
MAXLOOKUPMAP

Figure 5-4 DMMAXLOOKUPMAP

The DMCURRENCY object structure, as shown in Figure 5-5, supports currency migration.

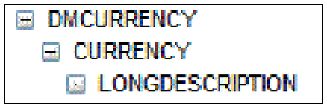

DMCURRENCY
CURRENCY
LONGDESCRIPTION

Figure 5-5 DMCURRENCY

The DMSETS object structure, as shown in Figure 5-6, supports item and company set migration.

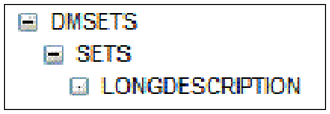

DMSETS
SETS
LONGDESCRIPTION

Figure 5-6 DMSETS

The DMORGANIZATION object structure, as shown in Figure 5-7, supports organization, address, and site migration.

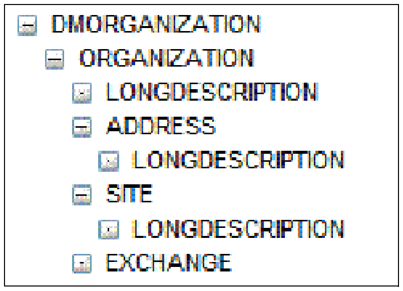
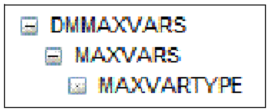

DMORGANIZATION
ORGANIZATION
LONGDESCRIPTION
ADDRESS
LONGDESCRIPTION
SITE
LONGDESCRIPTION
EXCHANGE

Figure 5-7 DMORGANIZATION

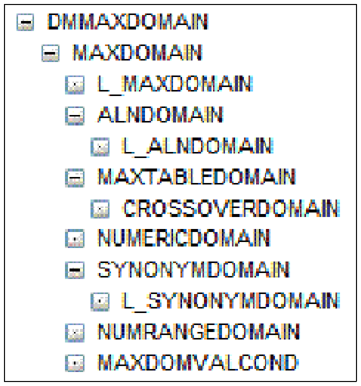
The DMMAXVARS object structure, as shown in Figure 5-8, supports the MAXVARS migration.



```
DMMAXVARS
  MAXVARS
    MAXVARTYPE
```

Figure 5-8 DMMAXVARS

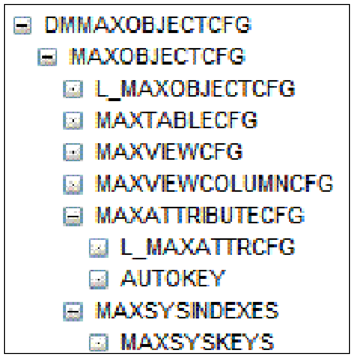
The DMMAXDOMAIN object structure, as shown in Figure 5-9, supports domain migration.



```
DMMAXDOMAIN
  MAXDOMAIN
    L_MAXDOMAIN
    ALNDOMAIN
      L_ALNDOMAIN
    MAXTABLEDOMAIN
      CROSSOVERDOMAIN
    NUMERICDOMAIN
    SYNONYMDOMAIN
      L_SYNONYMDOMAIN
    NUMRANGEDOMAIN
    MAXDOMVALCOND
```

Figure 5-9 DMMAXDOMAIN

The DMMAXOBJECTCFG object structure, as shown inFigure 5-10, supports table, view, column, attribute, and index migration.



```
DMMAXOBJECTCFG
  MAXOBJECTCFG
    L_MAXOBJECTCFG
    MAXTABLECFG
    MAXVIEWCFG
    MAXVIEWCOLUMNCFG
    MAXATTRIBUTECFG
      L_MAXATTRCFG
      AUTOKEY
    MAXSYSINDEXES
      MAXSYSKEYS
```

Figure 5-10 DMMAXOBJECTCFG

The DMMAXSEQUENCE object structure, as shown in Figure 5-11, supports sequence migration.

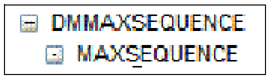


Figure 5-11 DMMAXSEQUENCE

The DMMAXRELATIONSHIP object structure, as shown in Figure 5-12, supports relationship migration.

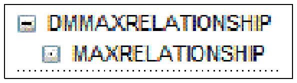


Figure 5-12 DMMAXRELATIONSHIP

The DMMAXINTOBJECT object structure, as shown in Figure 5-13, supports integration object migration.

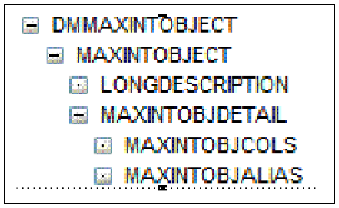


Figure 5-13 DMMAXINTOBJECT

The DMCONDITION object structure, as shown in Figure 5-14, supports conditional expression migration.

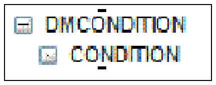


Figure 5-14 DMCONDITION

The DMGLCONFIGURE object structure, as shown in Figure 5-15, supports general ledger component migration.

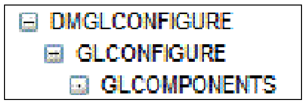


Figure 5-15 DMGLCONFIGURE

5.2.3 Migration groups

The object structures that are listed in 5.2.2, “Object structures” on page 98 are part of the DATA DICTIONARY migration group. Because the Data Dictionary migration group is the first group to be imported in any set of migration packages, the predefined package can be used, or a copy can be made and modified specifically to the configurations to migrate.

Figure 5-16 shows the DATA DICTIONARY group.

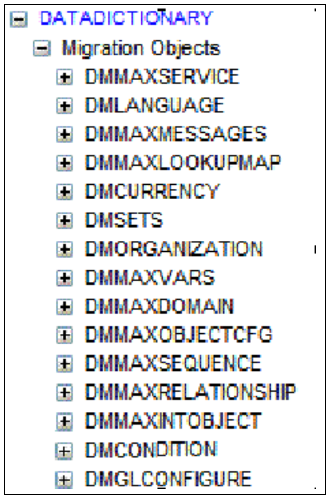
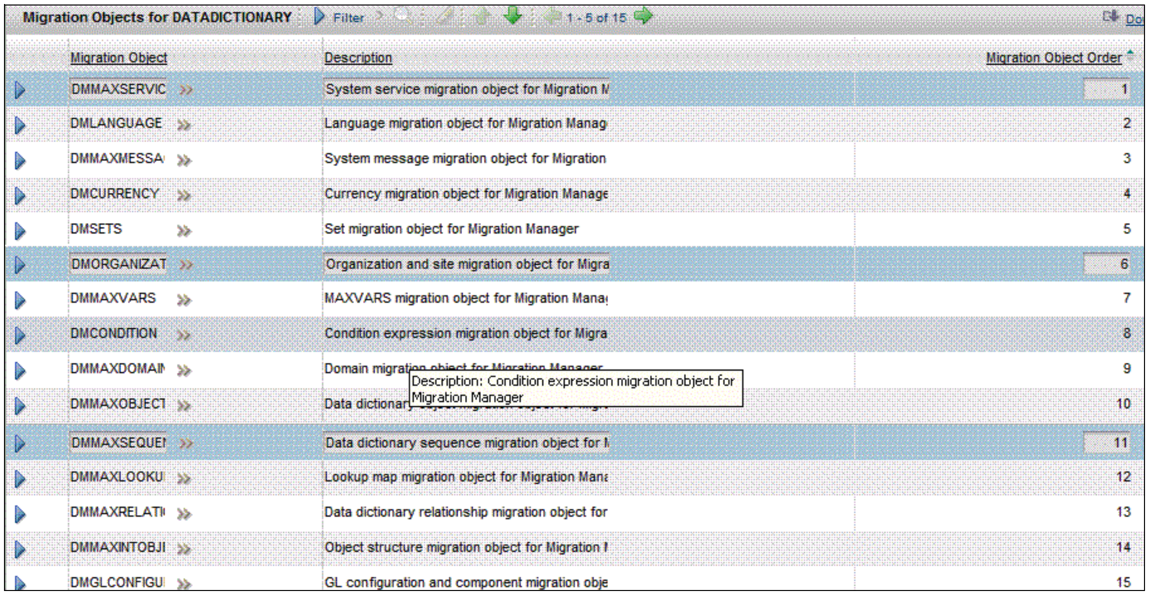


Figure 5-16 Data Dictionary group

There are two methods for limiting the data that is brought over by the migration package: the first method is to duplicate the initial package; the second method is to remove those object structures from the predefined package. The second method is used in this scenario (for more information, see 5.2.4, “Package definition” on page 103).

One of the features of the Migration Manager application is the ability to duplicate existing migration packages. This feature is helpful because multiple data dictionary packages are used to migrate separate functionality. Because the Data Dictionary package has no dependencies, the use of the duplicate functionality and the modifications to the Where clause in the package saves time in the process.

The order of the object structures within the migration group is shown in Figure 5-17.


Migration Objects for DATADICTIONARY   Filter   1 - 5 of 15

Migration Object	Description	Migration Object Order
DMMAXSERVIC	System service migration object for Migration M	1
DMLANGUAGE	Language migration object for Migration Manag	2
DMMAXMESSA	System message migration object for Migration	3
DMCURRENCY	Currency migration object for Migration Manage	4
DMSETS	Set migration object for Migration Manager	5
DMORGANIZAT	Organization and site migration object for Migra	6
DMMAXVARS	MAXVARS migration object for Migration Mana	7
DMCONDITION	Condition expression migration object for Migra	8
DMMAXDOMAIN	Domain migration object for Migration Manager	9
DMMAXOBJECT	Data dictionar	10
DMMAXSEQUEN	Data dictionary sequence migration object for M	11
DMMAXLOOKU	Lookup map migration object for Migration Mana	12
DMMAXRELATI	Data dictionary relationship migration object for	13
DMMAXINTOBJI	Object structure migration object for Migration M	14
DMGLCONFIGU	GL configuration and component migration obje	15

Description: Condition expression migration object for Migration Manager

Figure 5-17   Data Dictionary migration objects

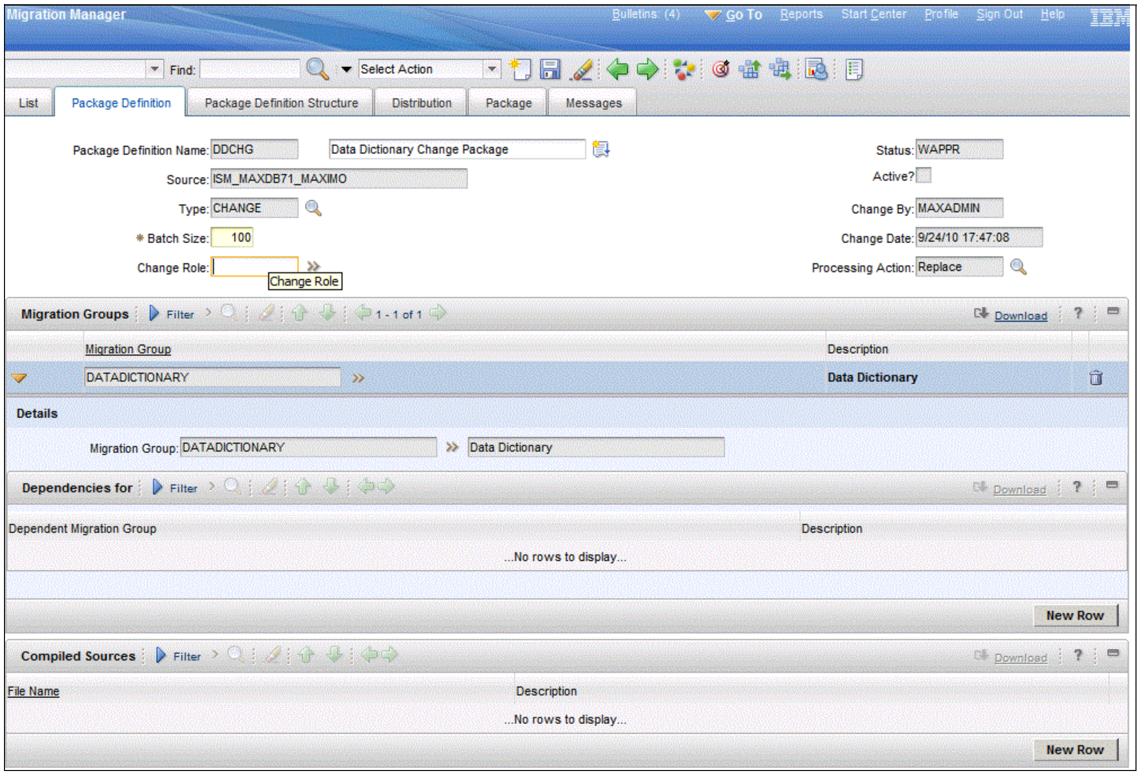
> **Easier viewing:** Figure 5-17 was modified to show all 15 objects in the object structure instead of creating three separate figures that show the default five objects in each structure.

5.2.4 Package definition

A new Change migration package is created for the Data Dictionary by using the predefined Data Dictionary migration package. We created this Change migration package before all of the configurations were built that are promoted from development to production as part of this scenario.

The Migration Manager application can be accessed from the Start Center by selecting **Go To** → **System Configuration** → **Migration** → **Migration Manager**.

The process for building the Change package is fairly straightforward, and it is performed by following the product documentation. Figure 5-18 shows the Package Definition tab of the new Change package.



Package Definition Name:	DDCHG — Data Dictionary Change Package
Source:	ISM_MAXDB71_MAXIMO
Type:	CHANGE
* Batch Size:	100
Change Role:	
Status:	WAPPR
Active?	
Change By:	MAXADMIN
Change Date:	9/24/10 17:47:08
Processing Action:	Replace

Migration Group	Description
DATADICTIONARY	Data Dictionary

Details

Migration Group: DATADICTIONARY — Data Dictionary

Dependent Migration Group	Description

...No rows to display...

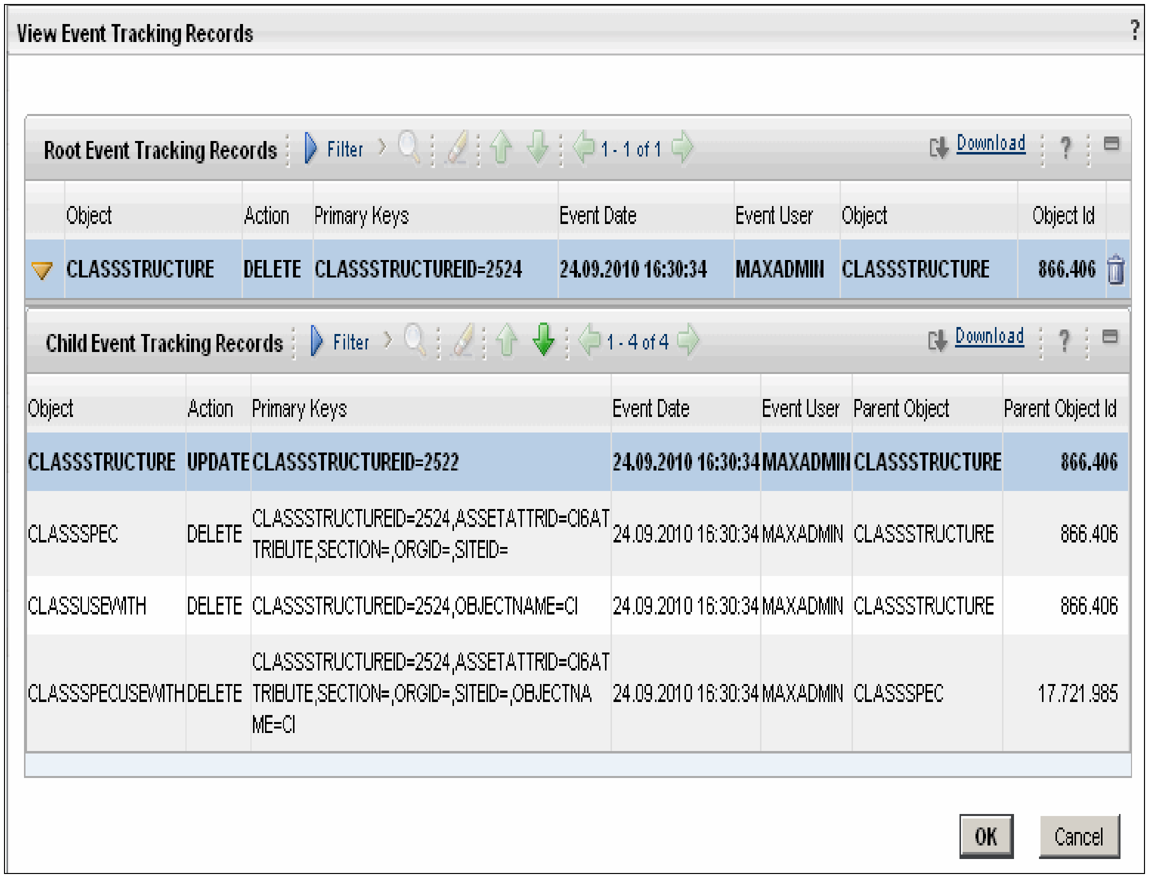
File Name	Description

...No rows to display...

Figure 5-18 Data Dictionary change package

The package is approved by using the Change Status toolbar or Select Action menu item and is activated by using the Activate Package Definition Select Action menu item. After the new package is active, the Change package tracks data dictionary changes.

Accessing the Change package after the configurations are built to migrate and selecting View Event Tracking Records shows the new configurations, as shown in Figure 5-19.



Root Event Tracking Records

Object	Action	Primary Keys	Event Date	Event User	Object	Object Id
CLASSSTRUCTURE	DELETE	CLASSSTRUCTUREID=2524	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406

Child Event Tracking Records

Object	Action	Primary Keys	Event Date	Event User	Parent Object	Parent Object Id
CLASSSTRUCTURE	UPDATE	CLASSSTRUCTUREID=2522	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406
CLASSSPEC	DELETE	CLASSSTRUCTUREID=2524,ASSETATTRID=CI6ATTRIBUTE,SECTION=,ORGID=,SITEID=	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406
CLASSUSEWITH	DELETE	CLASSSTRUCTUREID=2524,OBJECTNAME=CI	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406
CLASSSPECUSEWITH	DELETE	CLASSSTRUCTUREID=2524,ASSETATTRID=CI6ATTRIBUTE,SECTION=,ORGID=,SITEID=,OBJECTNAME=CI	24.09.2010 16:30:34	MAXADMIN	CLASSSPEC	17.721.985

Figure 5-19 Data Dictionary Change Package Tracking Records

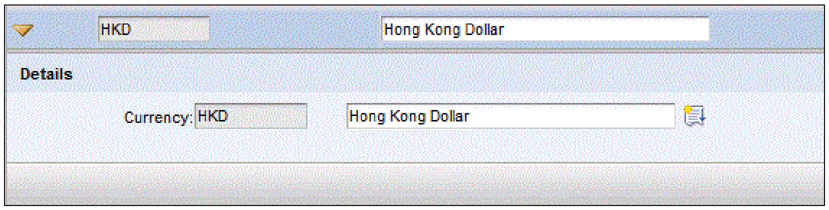
You now can select the items to promote, build, and distribute the package.

Three packages are built to migrate the data. The first package migrates the currency, organization, and sets. We separated this package from the other packages to show that multiple users can build and migrate separate sections of the data dictionary as required by their individual projects. This package does not include the migration of a new organization or clearing account. For more information, see 5.5, “Deployment considerations” on page 119.

The second and third packages are used to migrate the rest of the configurations. Two packages are required because of the inclusion of a crossover domain. For more information, see 5.5, “Deployment considerations” on page 119.

The first step for the migration to create the packages. This creation follows the standard process that is defined in the product documentation. It also is similar to the setup of the Change package with two exceptions: a package TYPE of SNAPSHOT that was created and the definition of the SQL that is used to define the package’s WHERE clauses. We describe these exceptions next.

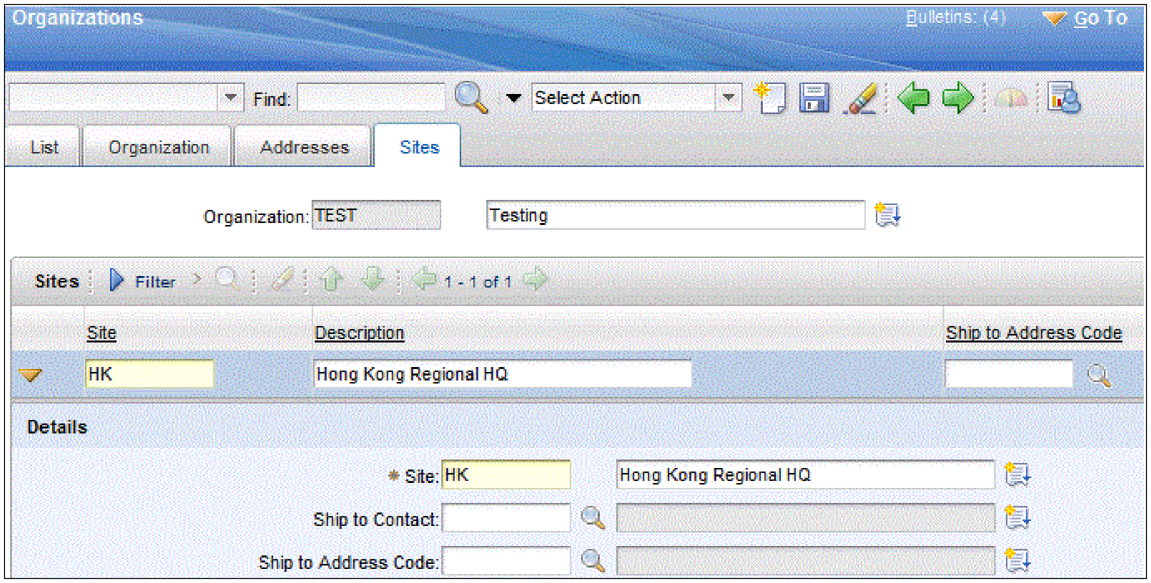
In the first package, you can obtain the names of the organization that are to be updated. You also can obtain and the sets and currency that are to be added in the event tracking of the initially created Change package or the documentation that was created for the project by the developer. These names are used in the SQL to define the CURRENCY, ORGANIZATION, and SET values. All other objects that are not migrated were set to 1=2. Figure 5-20, Figure 5-21 on page 107, and Figure 5-22 on page 107 show the data that is migrated in this package.


HKD Hong Kong Dollar

Details

Currency: HKD Hong Kong Dollar

Figure 5-20 New currency



Organization: TEST Testing

Sites Filter 1 - 1 of 1

Site	Description	Ship to Address Code
HK	Hong Kong Regional HQ	

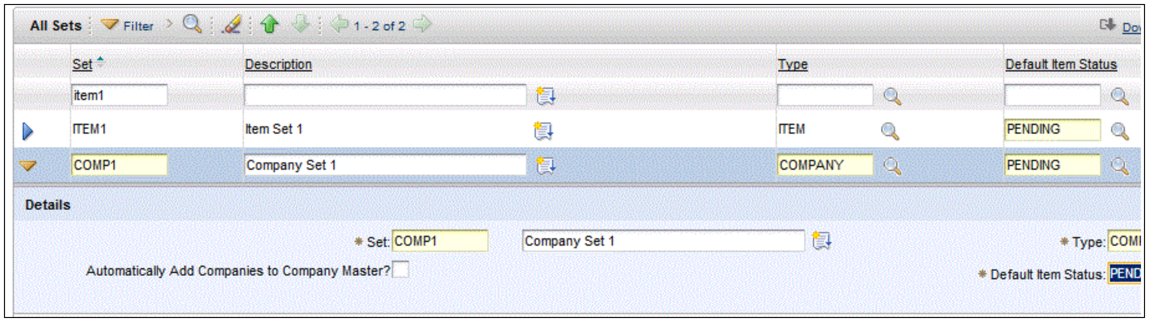
Details

* Site: HK Hong Kong Regional HQ

Ship to Contact:

Ship to Address Code:

Figure 5-21 New site for existing organization



Set	Description	Type	Default Item Status
item1			
ITEM1	Item Set 1	ITEM	PENDING
COMP1	Company Set 1	COMPANY	PENDING

Details

* Set: COMP1 Company Set 1

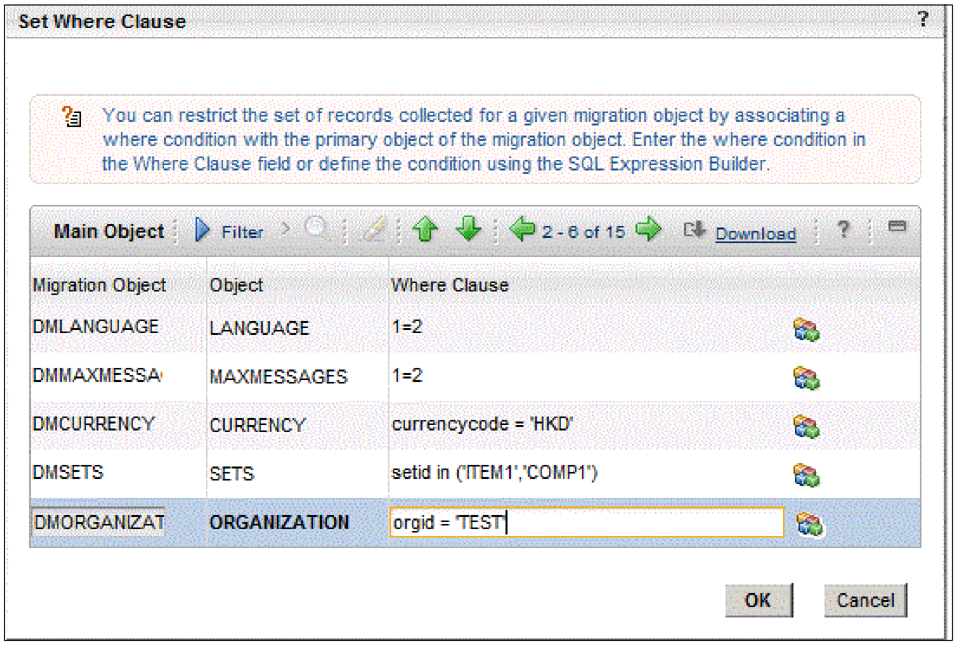
* Type: COMI

Automatically Add Companies to Company Master?

* Default Item Status: PEND

Figure 5-22 New item set and company set

The names are used in the SQL to define the CURRENCY, ORGANIZATION, and SET values. All other objects that are not migrated are set to 1=2. The column names that are used in the Where clause for each object can be found in the Change Package Event Tracking or in the field help for each update in its application, as shown in Figure 5-23.


Set Where Clause

You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

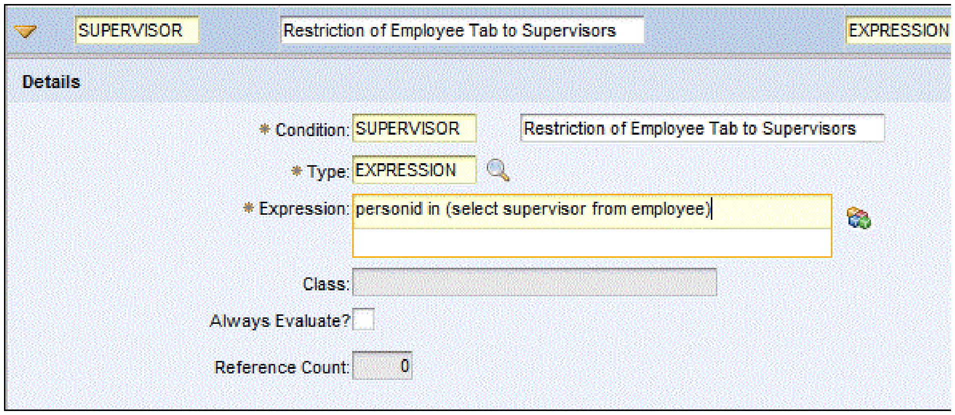
Migration Object	Object	Where Clause
DMLANGUAGE	LANGUAGE	1=2
DMMAXMESSA	MAXMESSAGES	1=2
DMCURRENCY	CURRENCY	currencycode = 'HKD'
DMSETS	SETS	setid in ('ITEM1','COMP1')
DMORGANIZAT	ORGANIZATION	orgid = 'TEST'

Figure 5-23 First Snapshot Package Where clause

After the SQL criteria are updated and all of the fields are defined, the package is saved.

The other packages can be created from the recently created Snapshot package by using the Duplicate Package Definition option in the Select Action menu. This option duplicates all of the information in the existing package except for the Package Definition Name and description. You must enter this information manually.

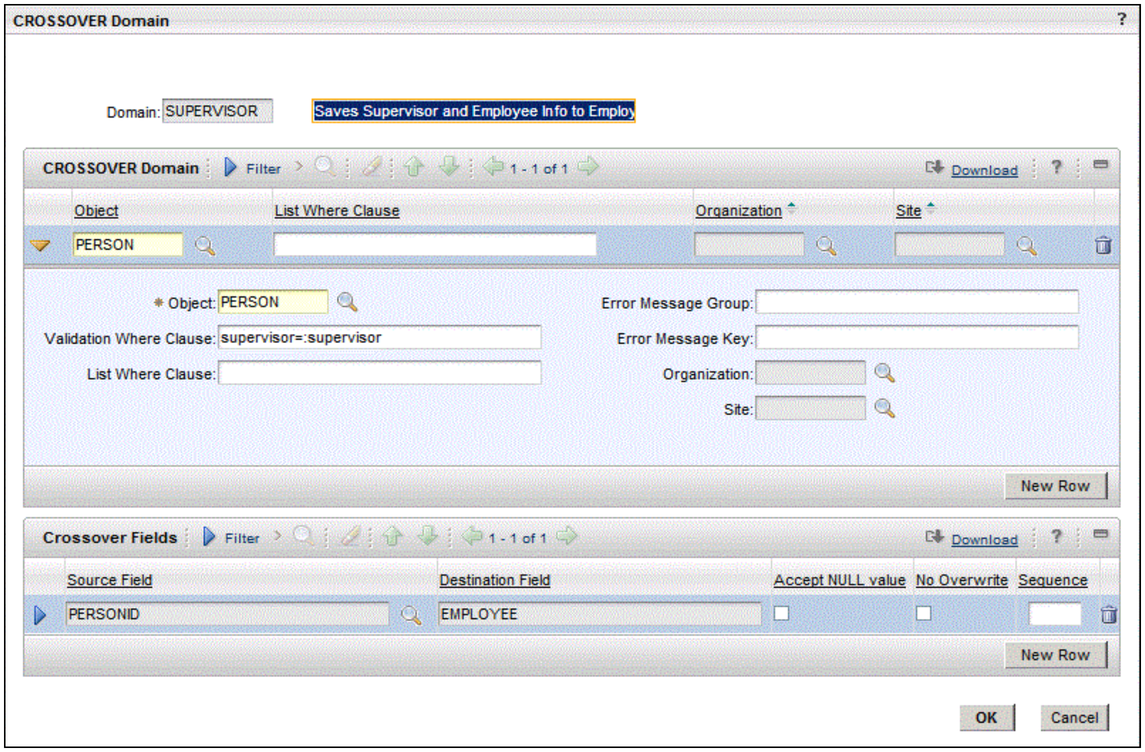
The SQL in the Where clauses for the second package is updated to migrate all of the other objects except for the DOMAIN, as described in 5.5.2, “Crossover and table domains” on page 119. Before the second package deployment, the change to the object structure is made and it is reverted before the third package deployment. Figure 5-24, Figure 5-25 on page 109, Figure 5-26 on page 110, Figure 5-27 on page 110, Figure 5-28 on page 111, Figure 5-29 on page 111, Figure 5-30 on page 112, and Figure 5-31 on page 113 show the objects that are migrated in this package.


SUPERVISOR Restriction of Employee Tab to Supervisors EXPRESSION

Details

* Condition:	SUPERVISOR — Restriction of Employee Tab to Supervisors
* Type:	EXPRESSION
* Expression:	personid in (select supervisor from employee)
Class:	
Always Evaluate?	☐
Reference Count:	0

Figure 5-24 New conditional expression


CROSSOVER Domain

Domain: SUPERVISOR Saves Supervisor and Employee Info to Employ

CROSSOVER Domain — Filter — 1 - 1 of 1 — Download

Object	List Where Clause	Organization	Site
PERSON			

* Object:	PERSON	Error Message Group:	
Validation Where Clause:	supervisor=:supervisor	Error Message Key:	
List Where Clause:		Organization:	
		Site:	

New Row

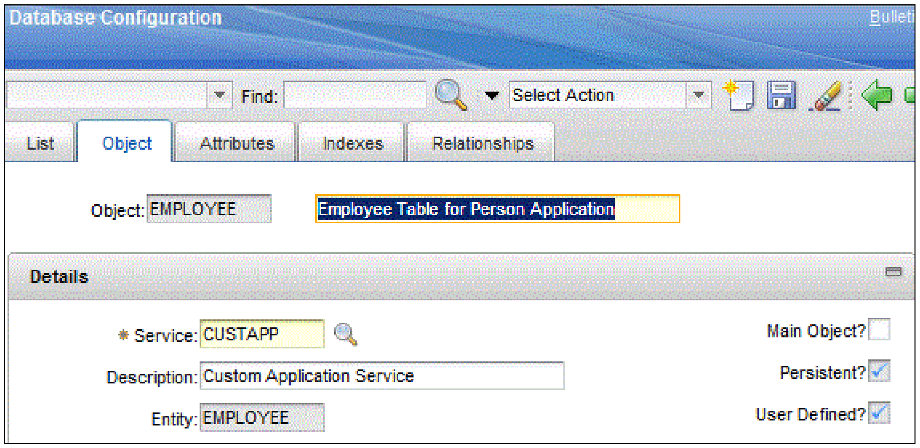
Crossover Fields — Filter — 1 - 1 of 1 — Download

Source Field	Destination Field	Accept NULL value	No Overwrite	Sequence
PERSONID	EMPLOYEE	☐	☐	

New Row

OK Cancel

Figure 5-25 New CROSSOVER Domain


Database Configuration

Find: Select Action

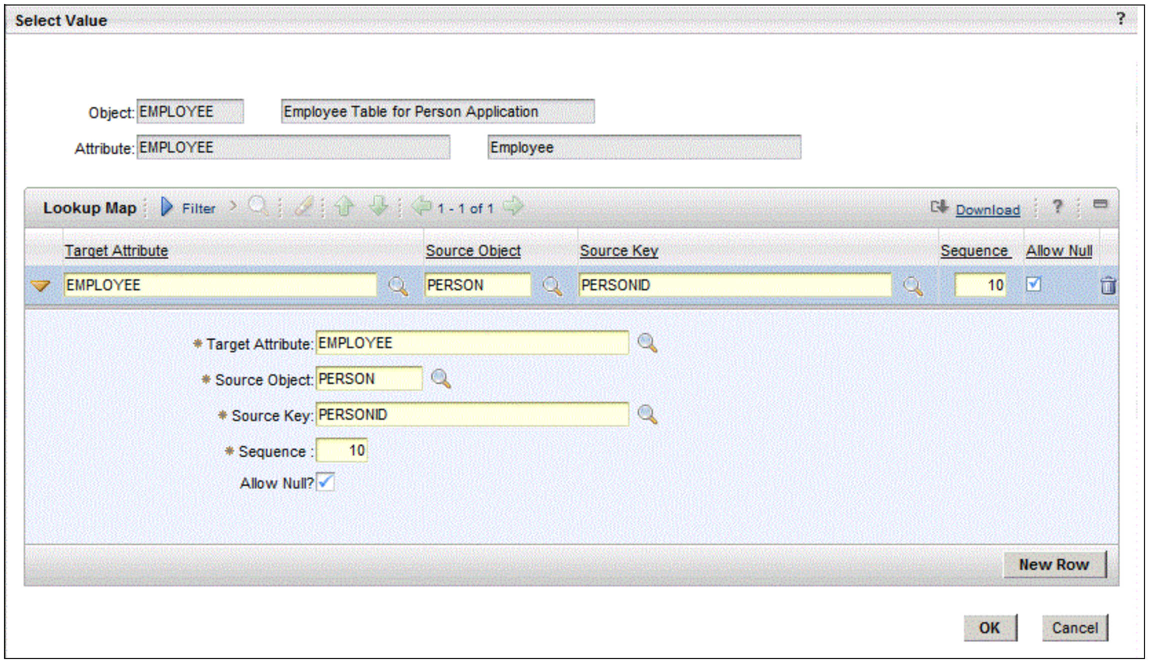
List | Object | Attributes | Indexes | Relationships

Object: EMPLOYEE Employee Table for Person Application

Details

* Service:	CUSTAPP	Main Object?	
Description:	Custom Application Service	Persistent?	✓
Entity:	EMPLOYEE	User Defined?	✓

Figure 5-26 New EMPLOYEE object


Select Value

Object: EMPLOYEE Employee Table for Person Application

Attribute: EMPLOYEE Employee

Lookup Map Filter 1 - 1 of 1 Download

Target Attribute	Source Object	Source Key	Sequence	Allow Null
EMPLOYEE	PERSON	PERSONID	10	✓

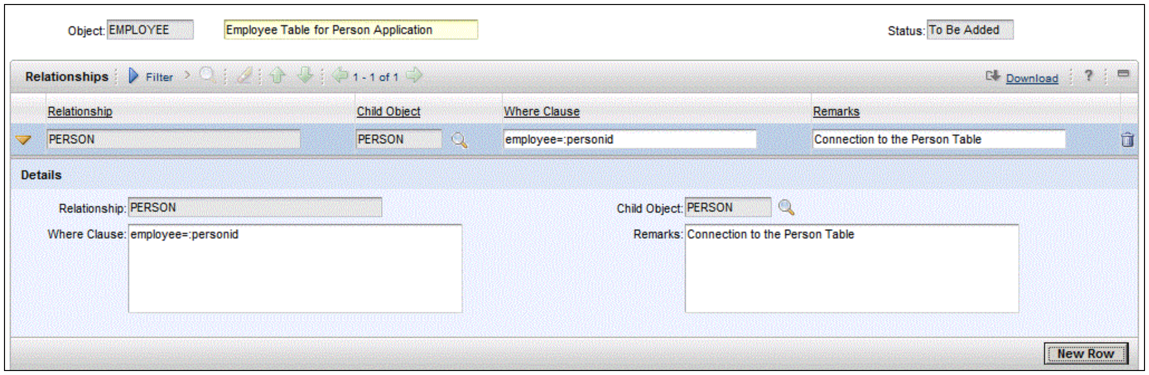
* Target Attribute: EMPLOYEE
* Source Object: PERSON
* Source Key: PERSONID
* Sequence : 10

Allow Null? ✓

New Row

OK Cancel

Figure 5-27 New Lookup Map


Object: EMPLOYEE Employee Table for Person Application Status: To Be Added

Relationships Filter 1 - 1 of 1 Download

Relationship	Child Object	Where Clause	Remarks
PERSON	PERSON	employee=:personid	Connection to the Person Table

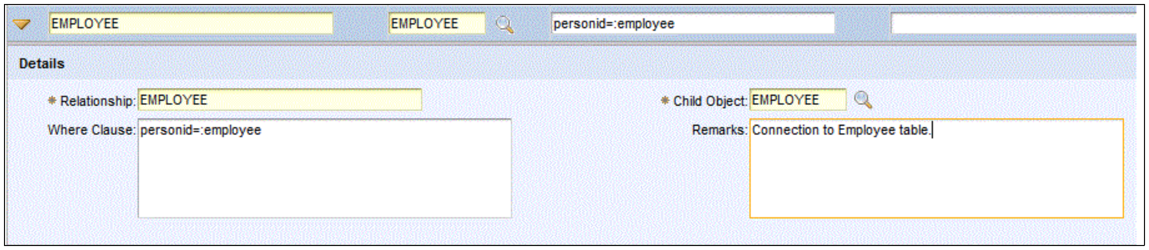
Details

Relationship: PERSON Child Object: PERSON

Where Clause: employee=:personid Remarks: Connection to the Person Table

New Row

Figure 5-28   New Relationship for EMPLOYEE object



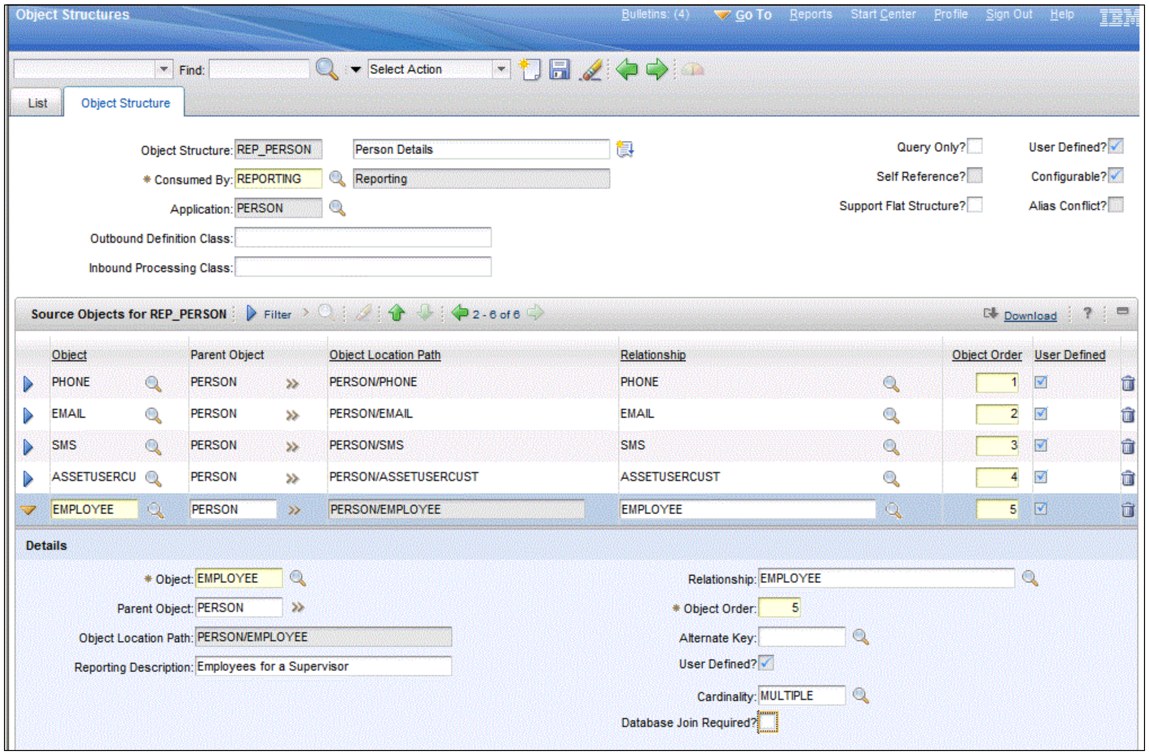
EMPLOYEE	EMPLOYEE	personid=:employee	

Details

* Relationship: EMPLOYEE * Child Object: EMPLOYEE

Where Clause: personid=:employee Remarks: Connection to Employee table.

Figure 5-29   New Relationship for Person object



Object Structure: REP_PERSON Person Details
* Consumed By: REPORTING Reporting
Application: PERSON
Outbound Definition Class:
Inbound Processing Class:

Query Only? ☐ User Defined? ✓
Self Reference? ☐ Configurable? ✓
Support Flat Structure? ☐ Alias Conflict? ☐

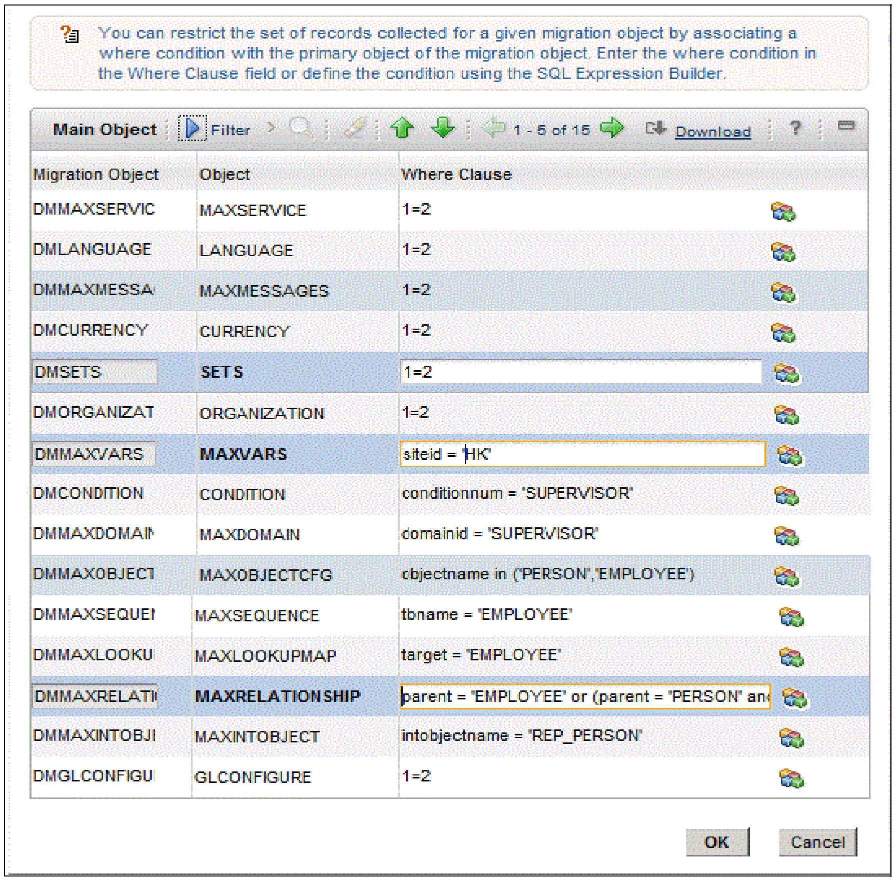
Source Objects for REP_PERSON Filter 2 - 6 of 6 Download

Object	Parent Object	Object Location Path	Relationship	Object Order	User Defined
PHONE	PERSON	PERSON/PHONE	PHONE	1	✓
EMAIL	PERSON	PERSON/EMAIL	EMAIL	2	✓
SMS	PERSON	PERSON/SMS	SMS	3	✓
ASSETUSERCU	PERSON	PERSON/ASSETUSERCUST	ASSETUSERCUST	4	✓
EMPLOYEE	PERSON	PERSON/EMPLOYEE	EMPLOYEE	5	✓

Details

* Object: EMPLOYEE
Parent Object: PERSON
Object Location Path: PERSON/EMPLOYEE
Reporting Description: Employees for a Supervisor
Relationship: EMPLOYEE
* Object Order: 5
Alternate Key:
User Defined? ✓
Cardinality: MULTIPLE
Database Join Required? ☐

Figure 5-30 New Join to EMPLOYEE object in existing PERSON report object structure


You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

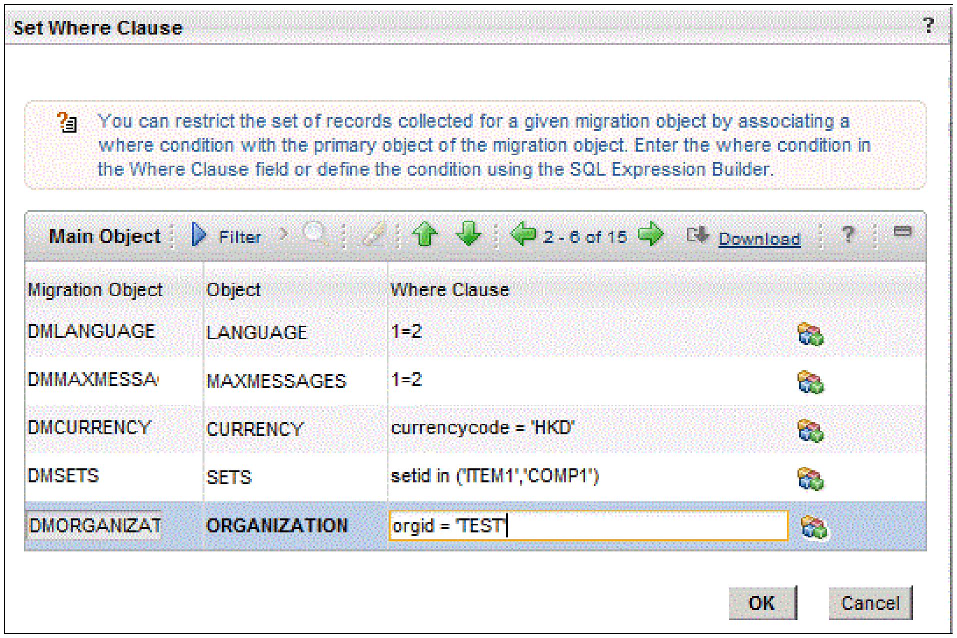
Migration Object	Object	Where Clause
DMMAXSERVIC	MAXSERVICE	1=2
DMLANGUAGE	LANGUAGE	1=2
DMMAXMESSA	MAXMESSAGES	1=2
DMCURRENCY	CURRENCY	1=2
DMSETS	SETS	1=2
DMORGANIZAT	ORGANIZATION	1=2
DMMAXVARS	MAXVARS	siteid = 'HK'
DMCONDITION	CONDITION	conditionnum = 'SUPERVISOR'
DMMAXDOMAIN	MAXDOMAIN	domainid = 'SUPERVISOR'
DMMAXOBJECT	MAXOBJECTCFG	objectname in ('PERSON','EMPLOYEE')
DMMAXSEQUEI	MAXSEQUENCE	tbname = 'EMPLOYEE'
DMMAXLOOKU	MAXLOOKUPMAP	target = 'EMPLOYEE'
DMMAXRELATI	MAXRELATIONSHIP	parent = 'EMPLOYEE' or (parent = 'PERSON' an
DMMAXINTOBJI	MAXINTOBJECT	intobjectname = 'REP_PERSON'
DMGLCONFIGU	GLCONFIGURE	1=2

Figure 5-31 Second Data Dictionary Snapshot package

Now that the SQL criteria that are related to the second package are updated, the packages can be saved.

After this package is saved, the third package is built from a duplicate of the first or second package. Again, the Package Definition Name and description are not duplicated and must be entered manually.

The SQL criteria for the final package contains only the Where clause for the MAXOBJECTCFG. After the object structure exclusion is reverted, the linkage between the object and the domain is created after it is deployed. All other objects are set to 1=2 to exclude them from the migration, as shown in Figure 5-32.



You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

Migration Object	Object	Where Clause
DMLANGUAGE	LANGUAGE	1=2
DMMAXMESSA	MAXMESSAGES	1=2
DMCURRENCY	CURRENCY	currencycode = 'HKD'
DMSETS	SETS	setid in ('ITEM1','COMP1')
DMORGANIZAT	ORGANIZATION	orgid = 'TEST'

Figure 5-32 Third Data Dictionary Snapshot package

Now that the SQL criteria for the final package are updated, it can be saved.

5.3 Solution B: Collection

By using collections, you can accomplish the same objectives as the use of the change package that is described in 5.2, “Solution A: Snapshot” on page 96. For this solution, we describe the steps that you must take to collect the same configurations into a new package that migrate those changes efficiently.

As described in Chapter 3, “Migration collections” on page 49, it is possible to add configurations through various methods. In this chapter, we use only the New Row button for illustrative purposes.

5.3.1 Define collection

From the system’s navigation menu, click **Go To** → **System Configuration** → **Migration** → **Migration Collections**

Enter a new record by clicking the **New Record** icon. Enter the name of the collection with an appropriate description. Now start entering configurations by clicking the **New Row** icon in the Configurations Table.

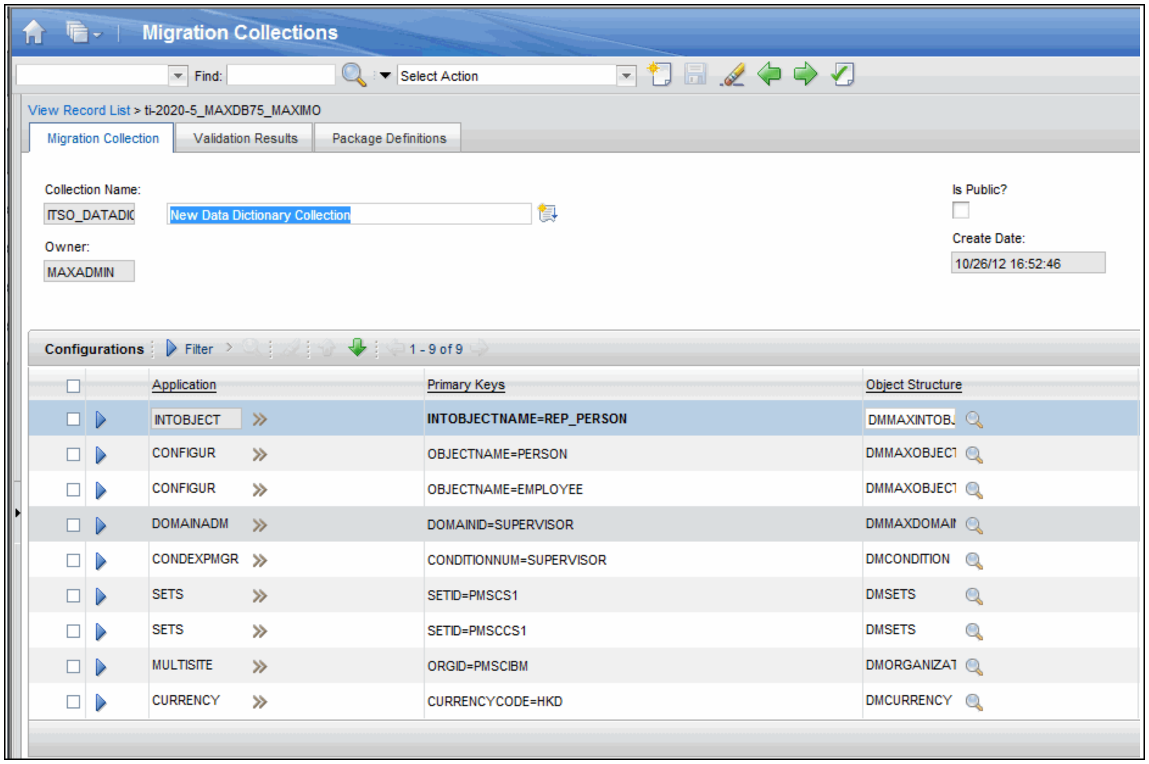
For each configuration, you enter the application from which the configuration originates in the new row. Enter the application by clicking the **Detail Menu** icon. Then, click the **Go To Application** icon to the right of the dialog box.

Important: This method is the only way that you can enter a configuration record with this method. Also, if you have more than one configuration for each application, you must create that entry in the collection separately.

The values for this collection are the same configurations that are used in 5.2, “Solution A: Snapshot” on page 96. The following configuration changes were made:

- Create new siteid
- Create new currency
- Create new item set and company set (optional)
- Create new condition
- Create new crossover domain
- Create new object
- Create new relationship
- Update reporting object structure with new object

The specific values for the changes are found in step on page 106 through step on page 114. After the entries that are found in the preceding notes section are reviewed, you notice that these entries are the specific objects that you add to the collection, as shown in Figure 5-33.



Collection Name: ITSO_DATADIC New Data Dictionary Collection

Owner: MAXADMIN

Is Public?

Create Date: 10/26/12 16:52:46

Application	Primary Keys	Object Structure
INTOBJECT	INTOBJECTNAME=REP_PERSON	DMMAXINTOBJ
CONFIGUR	OBJECTNAME=PERSON	DMMAXOBJECT
CONFIGUR	OBJECTNAME=EMPLOYEE	DMMAXOBJECT
DOMAINADM	DOMAINID=SUPERVISOR	DMMAXDOMAI
CONDEXPMGR	CONDITIONNUM=SUPERVISOR	DMCONDITION
SETS	SETID=PMSCS1	DMSETS
SETS	SETID=PMSCCS1	DMSETS
MULTISITE	ORGID=PMSCIBM	DMORGANIZAT
CURRENCY	CURRENCYCODE=HKD	DMCURRENCY

Figure 5-33 Data dictionary collection initial definition

In the next section, you notice that in the validation process, other configuration items are identified for inclusion.

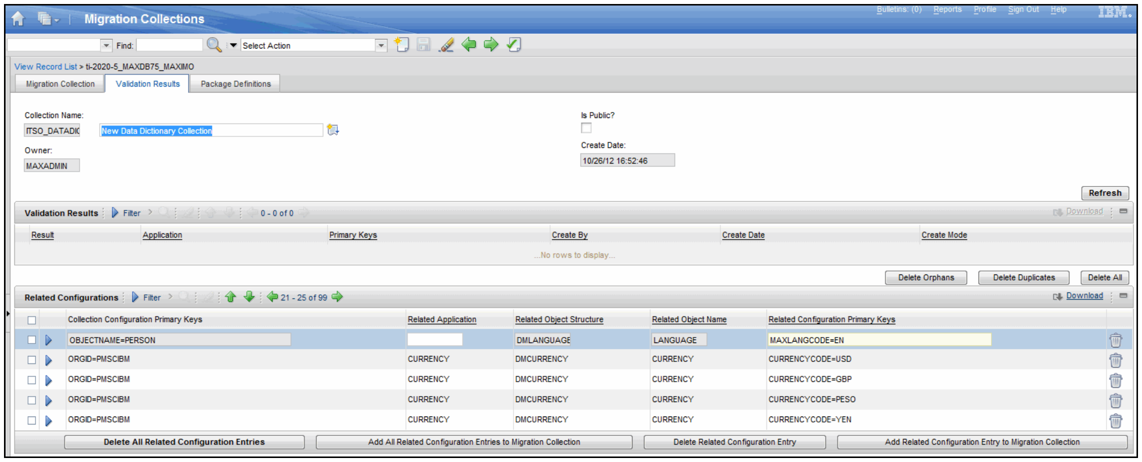
5.3.2 Validate collection

Now that the collection is defined, you must validate it. It is critical that you properly tune your validation rules according to what you require, as noted in 3.4.1, “Validation results” on page 56.

To validate, click **Select Action** → **Validate Migration Collection**

Now you can review the related validation entries by clicking the Validation Results tab and choose to add or delete the various configuration entries that are listed.

Figure 5-34 shows related configurations that occur because of the validation rules. You must decide whether you need these configurations as part of your collection.



View Record List > ti-2020-5_MAXDB75_MAXIMO

Collection Name: ITSO_DATADIC New Data Dictionary Collection

Owner: MAXADMIN

Is Public? ☐

Create Date: 10/26/12 16:52:46

Validation Results 0 - 0 of 0

Result	Application	Primary Keys	Create By	Create Date	Create Mode

...No rows to display...

Related Configurations 21 - 25 of 99

Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
OBJECTNAME=PERSON		DMLANGUAGE	LANGUAGE	MAXLANGCODE=EN
ORGID=PMSCIBM	CURRENCY	DMCURRENCY	CURRENCY	CURRENCYCODE=USD
ORGID=PMSCIBM	CURRENCY	DMCURRENCY	CURRENCY	CURRENCYCODE=GBP
ORGID=PMSCIBM	CURRENCY	DMCURRENCY	CURRENCY	CURRENCYCODE=PESO
ORGID=PMSCIBM	CURRENCY	DMCURRENCY	CURRENCY	CURRENCYCODE=YEN

Figure 5-34 Migration collection validation window that shows related configurations

After you decide on the entries, create the package by clicking **Select Action** → **Create Package Definitions**. Review the package definitions.

5.3.3 Package definition

From the package definition tab, you can browse to the Migration manager application for deployment.

5.4 Deployment

Tip: When significant structural changes are made to the database, a best practice is to make a backup before new migration packages are imported.

After the packages are saved, they can be exported from the Source system. The crossover domain migration requires the extra steps that are described in 5.5, "Deployment considerations" on page 119. Before the second package export, the DOMAINID must be excluded from the DMMAXOBJECTCFG object structure. Before the third package export, the DOMAINID must be included again in the DMMAXOBJECTCFG object structure.

Any data dictionary packages that make structural changes to the database require Administrator mode or a restart to complete the migration. In this example, structural changes are made. After Administrator mode is activated, the packages can be imported. For more information about the steps that are used for the deployment process, see the product documentation.

5.5 Deployment considerations

In the following sections, we describe the deployment considerations for this migration.

5.5.1 Organization and clearing accounts

The organization and the initial GL account, which is identified as the clearing account in the organization, are not migrated. The GL account must be associated with the organization, but the organization cannot be activated before the general ledger account is associated. The Migration Manager does not have the facility to activate the organization, so manual intervention is required to create the association and activate the organization. In the case of these items, it might be more efficient to manually enter the values in the new environment.

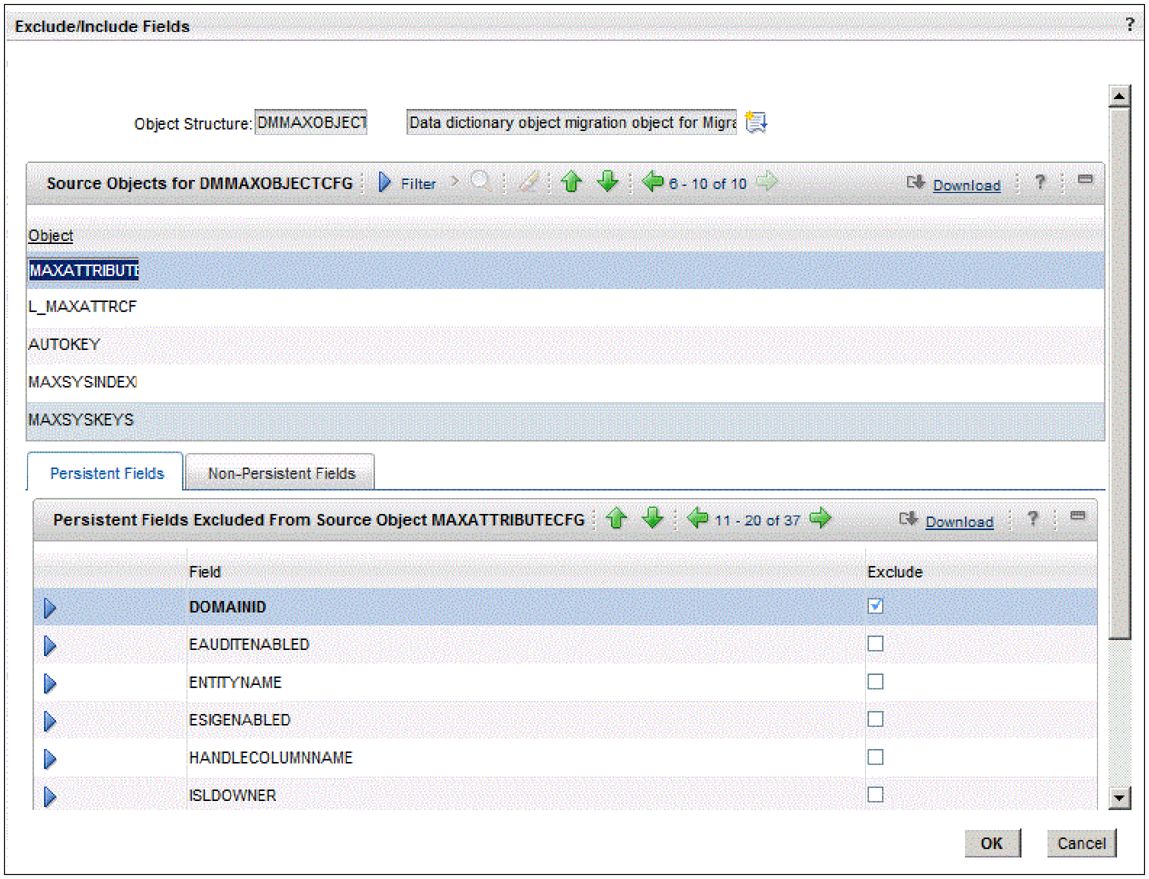
5.5.2 Crossover and table domains

Crossover and table domains present particular challenges when Migration Packages are used. Snapshot packages create a circular reference problem when they include these types of domains. This problem is caused when a crossover and table domain is referenced in an object that requires the domain to be present. However, domains always are migrated after objects in the data dictionary, thus the circular reference problem occurs when the object fails to find the referenced domain.

This process requires two packages: one package to contain the domain, the other to contain the reference. We build and deploy the following packages:

- Build, export, and import into the new environment the first package with no domain reference. The domain reference can be temporarily excluded from the object structure DMMAXOBJECTCFG in the source environment by opening the object structure and excluding the DOMAINID field from the MAXATTRIBUTECFG by using the Exclude/Include Fields window.
- Build, export, and import into the new environment the second package with the domain reference. The domain reference can be added again by following the reverse of the previous steps that were used to exclude the DOMAINID field from the first package. This process creates the correct link between the attribute and the domain.

Figure 5-35 shows the exclusion or inclusion that is required for crossover and table domain migration.


Exclude/Include Fields
Object Structure: DMMAXOBJECT
Data dictionary object migration object for Migra
Source Objects for DMMAXOBJECTCFG
Filter
6 - 10 of 10
Download
Persistent Fields
Non-Persistent Fields
Persistent Fields Excluded From Source Object MAXATTRIBUTECFG
11 - 20 of 37
Download
OK
Cancel

Object
MAXATTRIBUTE
L_MAXATTRCF
AUTOKEY
MAXSYSINDEX
MAXSYSKEYS

Field	Exclude
DOMAINID	☑
EAUDITENABLED	☐
ENTITYNAME	☐
ESIGENABLED	☐
HANDLECOLUMNNAME	☐
ISLDOWNER	☐

Figure 5-35 Exclusion or inclusion that is required for crossover and table domain migration

When collections is used, package exception rules automatically split the collection into two package definitions. The procedure to update the DMMAXOBJECTCFG object structure and build and deploy the two packages remains the same.

6

Migrating security configuration data

In this chapter, we describe the use case scenarios for security configuration data migration. Administrators perform security configuration by creating users, grouping them by security groups, and granting authorizations.

This chapter includes the following sections:

- Migrating a new security group
- Migrating the conditional user interface
- Migrating global data restrictions
- Migrating access definitions and LDAP information
- Considerations of security migrations

6.1 Migrating a new security group

System administrators configure the access that users can have to the system based on business requirements, including the following examples:

- Only users who are members of the PLANNING security group can use the Job Plans and Routes applications.
- Users who are members of the MAINTENANCE security group can initiate and complete work orders.
- Only users who are members of the FINANCIAL security group can access the Chart of Accounts application.
- Users who are members of the MAINTENANCE security group cannot see closed or canceled work orders.

The scenario that is presented in this chapter covers the migration of a new security group with new users, and the access definitions for existing application options. Additionally, this scenario includes the migration of conditions that are used for data restrictions.

To perform the tasks that we describe here, you must log in as an administrative user.

6.1.1 Requirements

The requirements for this migration consist of defining a new security group RBGROUP with a new user RBUSER. This group has full access to all of the options of the Work Order Tracking application, but members of the group have a data restriction that blocks any changes to work orders that are waiting on an approval status.

All of the configurations that are described here are created in the development environment and then promoted to the test and production environments.

6.1.2 Solution A: Snapshot

We define an approach to migrate a new security group and its Related Data in a single Snapshot migration package.

From a content perspective, the migration of a security group in this scenario also implies the migration of the following configurations in order:

- Conditions
- Security groups
- Users

Important: A security group might include a default Start Center template, a list of sites to which the group is authorized, and applications. It is not part of the requirements of this scenario to migrate sites, Start Center templates, and applications. We assume that these configurations already are migrated or exist in the target environment. For more information, see, “Deployment considerations” on page 129.

Configuration applications

The following sections show the data that was created as an example for this migration scenario.

Conditional Expression Manager

Conditions are SQL expressions or custom Java classes that are used, for example, to limit data access or editability. The restriction that the condition defines is only applied when the conditional expression evaluates to true. Conditions are created and managed by using the Conditional Expression Manager application, which is accessible by selecting **Go To** → **Administration** → **Conditional Expression Manager.**

For this scenario, we created a condition with the values that are shown in Table 6-1.

Table 6-1 New condition values

Field	Value
Condition	RBOOKCOND
Description	IBM Redbooks Condition
Type	EXPRESSION
Expression	:status = ‘WAPPR’
Always Evaluate	Selected

Security groups

You can create and manage security groups by using the Security Groups application, which is accessed by clicking **Go To** → **Security** → **Security Groups**.

For this scenario, we created a group with the values that are shown in Table 6-2.

Table 6-2 New security group values

Field	Value
Group	RBGROUP
Description	IBM Redbooks Group

Under the Sites tab, the **Authorize Group for All Sites** option is selected.

Under the Data Restrictions tab, we created an object restriction for the WORKORDER object. This data restriction, when evaluating to true, changes the editability of waiting on approval work orders to read-only. Table 6-3 shows the values of this new data restriction.

Table 6-3 New object restriction

Field	Field
Object	WORKORDER
Type	READONLY
Condition	RBOOKCOND

Important: You can use the same condition for multiple data restrictions. In this example, when you set the Object field to WORKORDER, you are creating this association.

You can associate the same condition with other objects if the condition can be applied to the other objects.

Under the Applications tab, the group received access to all of the options of the Work Order Tracking application.

Under the GL Components tab, we selected all of the component options.

Users

You can create and manage users with the Users application, which is accessed by clicking **Go To** → **Security** → **Users**.

For this scenario, we created a user with the values that are shown in Table 6-4.

Table 6-4 New user values

Field	Value
User	RBUSER1
User Name	RBUSER1
Type	TYPE 1

Important: Migration Security groups include the following important considerations:

- The Person field is required, and the system asks if you want to create a PERSON. It is not part of this scenario to migrate data that is created in the People application. We assume that the person that you choose exists in the target environment and that its status is ACTIVE.
- The product includes a predefined migration group that is called RESOURCES, which includes the DMPERSON object structure.
- The password field might be required. If so, you must enter a valid value, depending on your system configuration. We assume that you can create a user with all of the required data.

For more information about referenced data and password migration, see , “Deployment considerations” on page 129.

Under the Groups tab, we added the new group RBGROUP.

Object structures

The following object structures are delivered with the product to support the application security migration for this scenario:

- Figure 6-1 shows the DMCONDITION object structure in the DATADICTIONARY migration group that supports conditions.

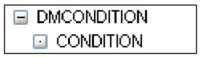
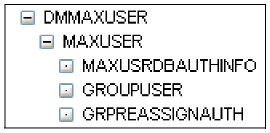


Figure 6-1 Condition object structure

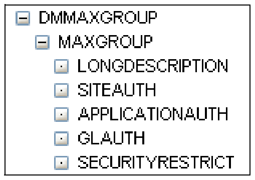
- Figure 6-2 shows the DMMAXUSER object structure in the APPSECURITY migration group that supports users.



```
DMMAXUSER
  MAXUSER
    MAXUSRDBAUTHINFO
    GROUPUSER
    GRPREASSIGNAUTH
```

Figure 6-2 User object structure

- Figure 6-3 shows the DMMAXGROUP object structure in the APPSECURITY migration group that supports the security groups.



```
DMMAXGROUP
  MAXGROUP
    LONGDESCRIPTION
    SITEAUTH
    APPLICATIONAUTH
    GLAUTH
    SECURITYRESTRICT
```

Figure 6-3 Security group object structure

Migration groups

The object structures that are listed in “Object structures” on page 125 are in two separate migration groups: DATADICTIONARY and APPSECURITY. The migration groups that are available in the product include dependencies on other migration groups, which are not required in this scenario.

A suitable and efficient approach to migrate the data that you created is to define a new migration group that contains only the object structures that were identified to achieve the migration requirements of this scenario.

You can create the migration group from the Migration Groups application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Migration Groups**.

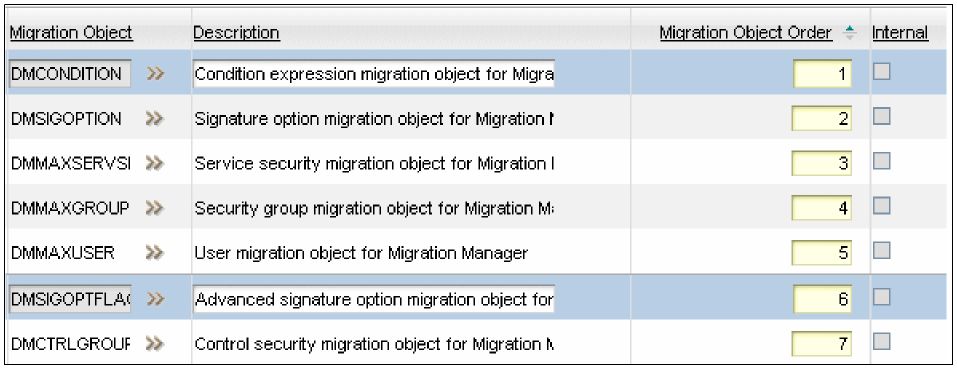
Tip: You can duplicate the existing APPSECURITY group and remove all of the dependencies.

Table 6-5 shows the values of the new migration group. All other fields use default values.

Table 6-5 New migration group

Field	Value
Migration Group	RBSECURITY01
Description	IBM Redbooks Security 01

Figure 6-4 shows the object structures that are in the new group and their order.



Migration Object	Description	Migration Object Order	Internal
DMCONDITION	Condition expression migration object for Migra	1	☐
DMSIGOPTION	Signature option migration object for Migration I	2	☐
DMMAXSERVSI	Service security migration object for Migration I	3	☐
DMMAXGROUP	Security group migration object for Migration M	4	☐
DMMAXUSER	User migration object for Migration Manager	5	☐
DMSIGOPTFLA(	Advanced signature option migration object for	6	☐
DMCTRLGROUF	Control security migration object for Migration M	7	☐

Figure 6-4 RBSECURITY01 object structures

Package definition

Now that the data set and the migration group are defined, the migration process can begin. The first step is to define the package by using the Migration Manager application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Migration Manager**.

A single Snapshot package is enough to migrate the security configurations that we made. The package must contain the migration group that you created in “Migration groups” on page 126. Table 6-6 shows the values of this package. All other fields use default values.

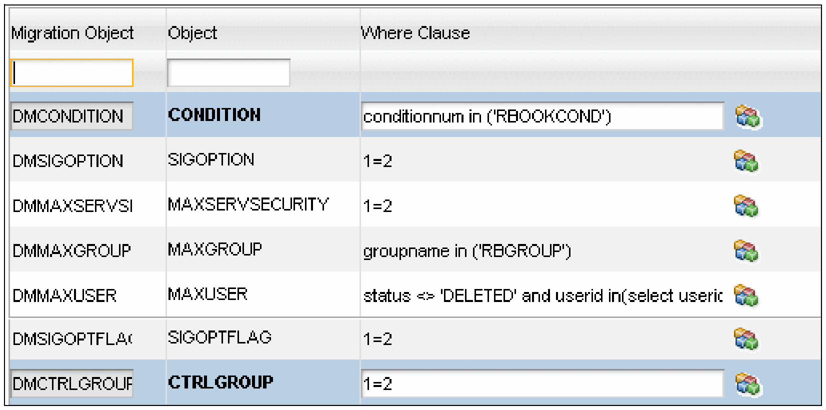
Table 6-6 RBSECPKG01 migration package values

Field	Value
Package Definition Name	RBSECPKG01
Description	IBM Redbooks Security Pkg 01
Type	SNAPSHOT
Processing Action	AddChange

Under the Migration Groups tab, we added the RBSECURITY01 migration group.

Defining SQL criteria

Figure 6-5 shows the queries that were created for each object structure that is part of the migration group.



Migration Object	Object	Where Clause
DMCONDITION	CONDITION	conditionnum in ('RBOOKCOND')
DMSIGOPTION	SIGOPTION	1=2
DMMAXSERVSI	MAXSERVSECURITY	1=2
DMMAXGROUP	MAXGROUP	groupname in ('RBGROUP')
DMMAXUSER	MAXUSER	status <> 'DELETED' and userid in(select useric
DMSIGOPTFLA(	SIGOPTFLAG	1=2
DMCTRLGROUF	CTRLGROUP	1=2

Figure 6-5 Object structures queries

The complete Where clause for the DMMAXUSER migration object is shown in the following example:

```
status <> 'DELETED' and userid in(select userid from groupuser where
groupname in ('RBGROUP'))
```

Tip: Certain queries are defined as 1=2. You do not need to create a migration group for other security migration-related scenarios. You must create other packages and define separate filters only.

If an incorrect SQL Where clause is entered, the Migration Manager application reports the BMXAA7010E error. You must correct the SQL Where clause before the package definition can be saved.

Best practice: You must always exclude users whose status is DELETED from your migration package. If those users are part of the migration package, the system reports the BMXAA3837E error when you try to deploy the package to the target environment.

You must also avoid migrating the MAXADMIN and MAXINTADM system users. If those users are part of the migration package, the system reports the BMXAA3829E error when you try to deploy the package to the target environment.

Deployment

After you save and approve the package definition, you can proceed with the creation, distribution, and deployment processes.

Deployment considerations

For this scenario, you must observe the important points that we describe in this section for your security configurations to work in the target environment in the same way that they worked in the source environment.

Dependencies

The migration that you performed depends on the following data that must be in the target environment:

- Administrator mode must be enabled in the target environment.
- You must migrate new applications, new objects, and changes to existing objects before the deployment.

  Best practice: Apply the same license keys in the target environment as in the source environment. The use of diverse licenses enables or disables separate sets of applications, which results in deployment errors.

- If you use a default Start Center template for your security groups, you must migrate or manually create the Start Center in the target environment before deployment. For more information about Start Center template migration, see Chapter 8, “Migrating applications and Start Centers” on page 205.
- You must migrate or manually create the sites to which your group is authorized in the target environment before deployment.
- The group that we created in this example authorizes the use of GL account components. You must migrate or manually create the GL account configuration in the target environment before deployment.
- The user that we created in this example was related to an existing person. You must migrate or manually create all of the related persons in the target environment before deployment. For more information about an approach to migrating person information, see Chapter 7, “Migrating escalations that include automation scripts” on page 163.

Application options: It is not part of the migration requirements of this scenario to migrate application options (sigoptions). This configuration uses the DMSIGOPTION object structure, which is available in the product and is part of the migration requirements that are described in 6.2, “Migrating the conditional user interface” on page 137.

Handling passwords

The Migration Manager application does not migrate passwords for security reasons. All migrated users’ passwords are and they receive an email message with a temporary password, which expires after the first login. For this reason, an email server must be configured in the target environment before the migration of the security package. All migrated users also must have a valid email address as part of their Person record.

Important: In case the email server is not configured or users do not receive their temporary password, if your system is not configured with Lightweight Directory Access Protocol (LDAP) or an auto-generated password, a user with administrative privileges can define temporary passwords manually.

For more information about an LDAP scenario, see 6.4, “Migrating access definitions and LDAP information” on page 154.

Attribute restrictions

You can use conditions to define data restrictions for objects and attributes. Attribute data restrictions are migrated in the same way that object restrictions are migrated. The difference is that the restrictions apply for an attribute and not for the whole object.

Data that is not migrated

The following security-related data is not migrated:

- Password hints (questions and answers)
- Login tracking
- Native database users
- Storeroom authorization
- Labor authorizations
- Limits and tolerances
- Status history
- Purchase GL

6.1.3 Solution B: Migration collection

The alternative solution for migrating Security Groups is to use a Migration Collection. For this solution, we describe the steps you must take to collect the same configurations into a new package that migrates those changes efficiently.

As described in Chapter 3, "Migration collections" on page 49, it is possible to add configurations through various techniques.

Define collection

By using the system's navigation menu, click **Go To** → **System Configuration** → **Migration** → **Migration Col**lections to go to the Migration Collections application. In this solution, we are using the Migration Collections application.

For this scenario, you add collection support to applications. You add this support so the applications can be used to add objects to migration collections. To complete this process, we use the Add migration Collection to Application option that is found on the Select Action menu, as shown in Figure 6-6.

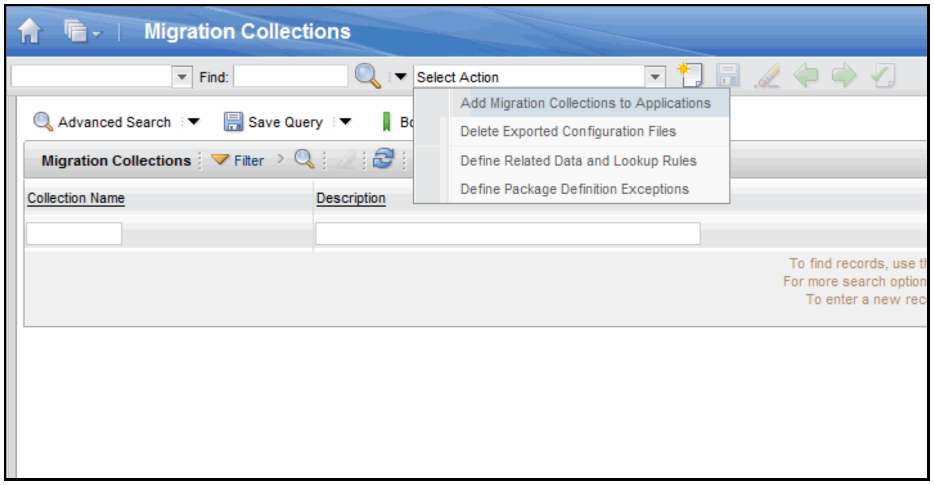
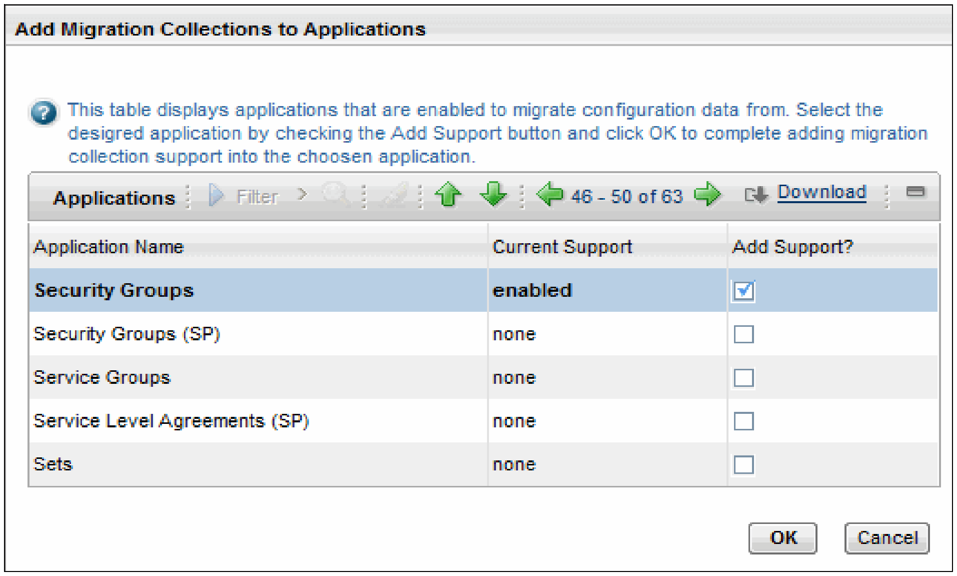


Figure 6-6   Adding migration collections to applications

This option opens a window in which you can select the applications from which you need configurations information. In this scenario, you use Security Groups, Users, and Conditional Expression Manager. Figure 6-7 shows one of the three enabled.

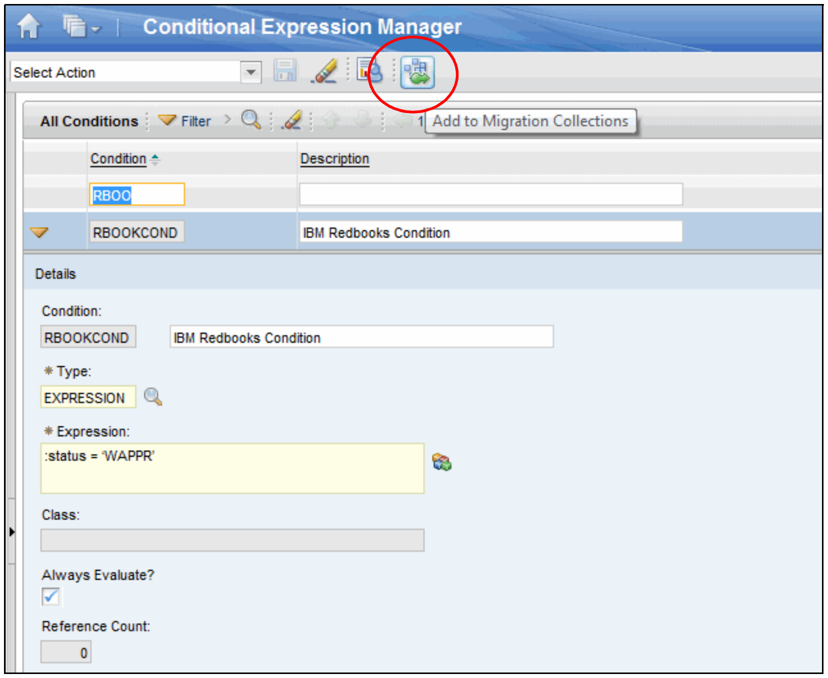


This table displays applications that are enabled to migrate configuration data from. Select the desigred application by checking the Add Support button and click OK to complete adding migration collection support into the choosen application.

Application Name	Current Support	Add Support?
Security Groups	**enabled**	✓
Security Groups (SP)	none	☐
Service Groups	none	☐
Service Level Agreements (SP)	none	☐
Sets	none	☐

Figure 6-7 Adding migration collection support to applications

Now that you set up collections support, you must next create a collection. Then, you can use the applications to add the configurations to the newly created collection. When you save the configuration, click **Add to Migration Collection** and the configuration is added to your new collection, as shown in Figure 6-8.



Condition	Description
RBOO	
RBOOKCOND	IBM Redbooks Condition

Condition: RBOOKCOND IBM Redbooks Condition

* Type: EXPRESSION

* Expression: :status = 'WAPPR'

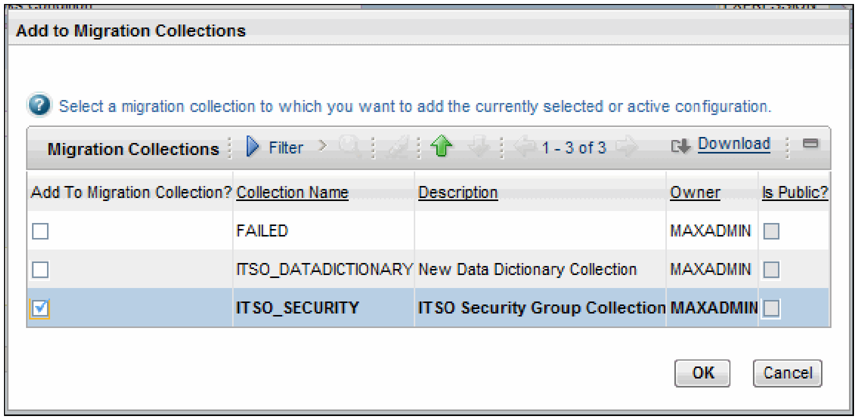
Class:

Always Evaluate? ✓

Reference Count: 0

Figure 6-8 Add to migration collection option highlighted

Select the wanted collection from the dialog box as shown in Figure 6-9. Then, click **OK**.


Add to Migration Collections

Select a migration collection to which you want to add the currently selected or active configuration.

Migration Collections | Filter | 1 - 3 of 3 | Download

Add To Migration Collection?	Collection Name	Description	Owner	Is Public?
☐	FAILED		MAXADMIN	☐
☐	ITSO_DATADICTIONARY	New Data Dictionary Collection	MAXADMIN	☐
✓	ITSO_SECURITY	ITSO Security Group Collection	MAXADMIN	☐

Figure 6-9 Select the ed collection

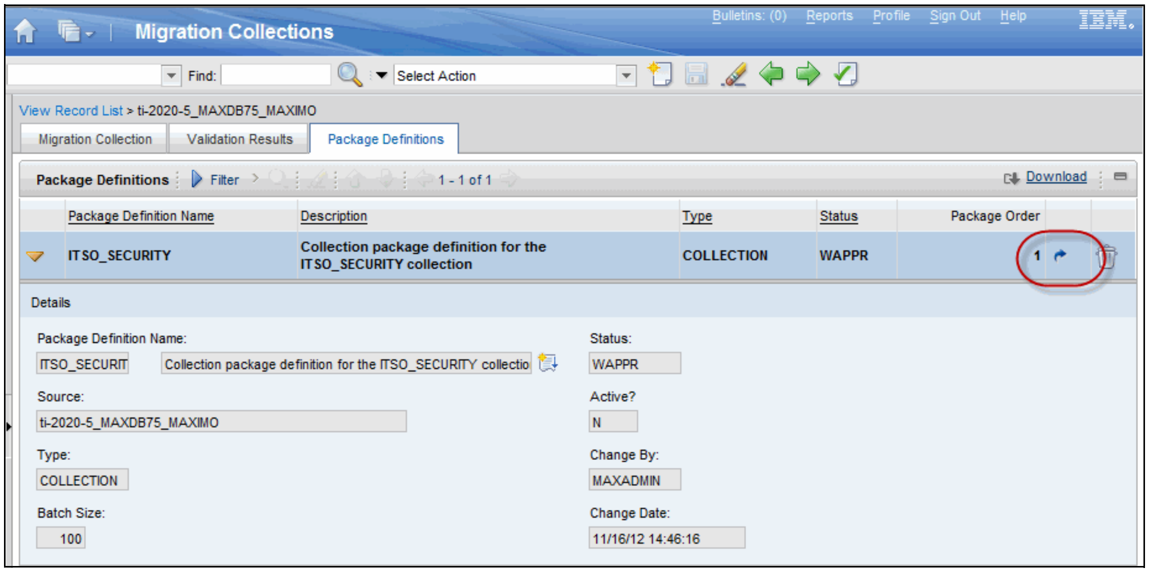
Repeat this process for each configuration that you want to include in your collection. When you are finished, review the collection to ensure that the wanted configurations are included in the collection.

Validate collection

Now that the collection is defined, you must validate it to ensure that you have all of the necessary configuration items that are required to properly migrate the collection. Click **Select Action** → **Validate Migration Collection.** After you validate the collection, review the validation configurations. In this scenario, another set of configurations regarding Signature Options, Max Menu, Max Relationships, and so on, are added as part of the validation. When you are satisfied with the review, add the validation entries that you need.

Package definition

Now click **Select Action** → **Create Package Definition.** From the package definition tab, you can browse to the Migration manager application for deployment by clicking the **Go To Migration Manager** icon, as shown in Figure 6-10.



Package Definition Name	Description	Type	Status	Package Order
ITSO_SECURITY	Collection package definition for the ITSO_SECURITY collection	COLLECTION	WAPPR	1

Details

Package Definition Name: ITSO_SECURIT Collection package definition for the ITSO_SECURITY collectio	Status: WAPPR
Source: ti-2020-5_MAXDB75_MAXIMO	Active? N
Type: COLLECTION	Change By: MAXADMIN
Batch Size: 100	Change Date: 11/16/12 14:46:16

Figure 6-10 Go to migration manager icon highlighted

Now you can use the Migration Manager application to change the status of the package to Approved status (APPR), then create and distribute the packaged collection.

6.2 Migrating the conditional user interface

As part of the product configurations, users can redefine an existing window, create new application options, and set conditions to show or hide user interface controls that are based on the following example business requirements:

- Hide the Failure Reporting tab of the Work Order Tracking application for users who are members of the ENGINEERS group and if the work order is for preventive maintenance.
- Hide the Actual Start and Actual Finish fields in the major tab of the Work Order Tracking application when work orders that are incomplete are displayed.

This scenario describes the migration of a new application option, new conditional user interface definitions, a new security group, and new access definitions.

Administrative user: To perform the tasks that we describe here, you must sign in as an administrative user.

6.2.1 Requirements

For this scenario, our requirement is to hide the Actual Start and Actual Finish fields when the user who is accessing the work order is a member of RBGROUP1 and the work order is incomplete. To configure this requirement, one approach is to define a new RBGROUP1 security group and a new option named RBOPTION1 for the Work Order Tracking application. The conditional user interface also must be configured to hide the fields by using a condition that is based on the work order status.

All of the configurations that we describe are created in a development environment and then promoted to the test and production environments.

6.2.2 Solution A: Snapshot

We define an approach so that a set of new application options, the conditional user interface, the new group, and the Related Data is migrated in a single Snapshot migration package.

The migration of the conditional user interface that is described in this scenario implies the migration of the following configurations in order:

- Conditions
- Sigoptions
- Security groups
- Control security

Configuration applications

The following sections show the configuration that was created as an example for this migration scenario.

Conditional Expression Manager

Conditions are either SQL expressions or custom Java classes that are used, for example, to limit data access or editability. The restriction that the condition defines is applied only when the conditional expression evaluates to true. Conditions are created and managed by using the Conditional Expression Manager application, which is accessible by selecting **Go To** → **Administration** → **Conditional Expression Manager**.

For this scenario, we created a condition with the values that are shown in Table 6-7.

Table 6-7 New condition values

Field	**Value**
Condition	RBOOKCOND1
Description	IBM Redbooks Condition 1
Type	EXPRESSION
Expression	:status = ‘COMP’
Always Evaluate	checked

Security groups

You can create and manage security groups by using the Security Groups application, which is accessed by clicking **Go To** → **Security** → **Security Groups**.

For this scenario, we created a group with the values that are shown in Table 6-8.

Table 6-8 New security group values

Field	Value
Group	RBGROUP1
Description	IBM Redbooks Group 1

Under the Sites tab, we select the **Authorize Group for All Sites** option.

Authorizing sites: If you do not want to authorize all sites, make sure that you already migrated the sites. For more information, see “Deployment considerations” on page 129.

Under the Applications tab, this group received access to all of the options of the Work Order Tracking application.

Under the GL Components tab, we selected all of the component options.

Application Designer

You can create and manage applications with the Application Designer application, which is accessed by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**.

After you open the application WOTRACK, you can create an application option by clicking **Select Action** → **Add/Modify Signature Option**.

For this scenario, we created the application option in Table 6-9 for the Work Order Tracking application (WOTRACK).

Table 6-9 New sigoptions

Option	Description	Advanced signature options
RBOPTION1	IBM Redbooks Opt1	None

To configure a conditional property for a user interface control, you must select the control first and then click **Control Properties** in the Application Designer toolbar, as shown in Figure 6-11.

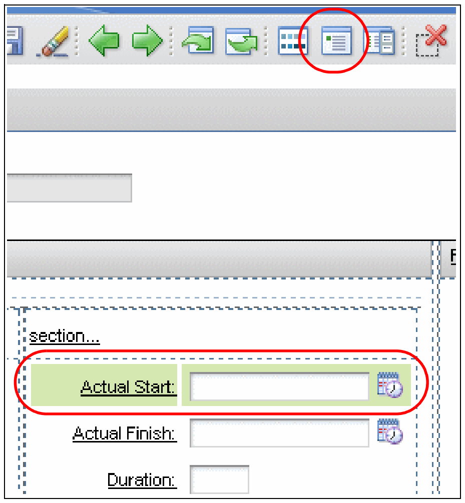


Figure 6-11 Control properties access

We needed to add conditional configuration to the Actual Start and Actual Finish fields. Therefore, we selected the first field in the Work Order tab and clicked the Control Properties toolbar. You can control the conditional configuration by clicking **Configure Conditional Properties**. The configuration that we created for this field is shown in Table 6-10.

Table 6-10 New control security condition for Actual Start field

Field or table	**Value**
Signature Option	RBOPTION1
Security Groups	RBGROUP1
Conditions for security group RBGROUP1	RBOOKCOND1
Property values when condition is true	display - true
Property values when condition is false	display - false

Then, we follow the same steps for the second field, by using the same values that are shown in Table 6-10.

We returned to the Security Groups application to grant access to RBOPTION1 to group RBGROUP1. The access definitions must be migrated as described in the migration requirements.

Object structures

The following object structures are delivered with the product to support the conditional user interface migration for this scenario.

The DMSIGOPTION object structure in the APPSECURITY migration group supports the application options, as shown in Figure 6-12.

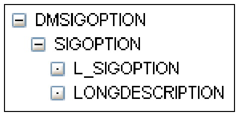


Figure 6-12 DMSIGOPTION object structure

The DMCTRLGROUP object structure in the APPSECURITY migration group supports control security, as shown in Figure 6-13.

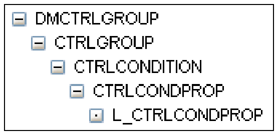


Figure 6-13 DMCTRLGROUP object structure

For more information about the DMCONDITION and DMMAXGROUP object structures, which are required by this scenario, see, “Object structures” on page 125.

Migration groups

The object structures that are listed in “Object structures” on page 141 are contained in two separate migration groups. The migration groups that are available in the product include dependencies on other migration groups, which are not required in the current scenario.

A suitable and efficient approach to migrate the configuration that you created is to define a new migration group that contains only the object structures that are identified for this scenario. The migration group equals the group that we created in “Migration groups” on page 126.

Package definitions

You now have both the data set and the migration group defined, and the migration process can begin. The first step is to define the package by using the Migration Manager application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Migration Manager**.

A single Snapshot package is enough to migrate the conditional user interface configurations that we created. The interface must contain the migration group that we created in , "Migration groups" on page 141. Table 6-11 shows the values of this package. All of the other fields feature default values.

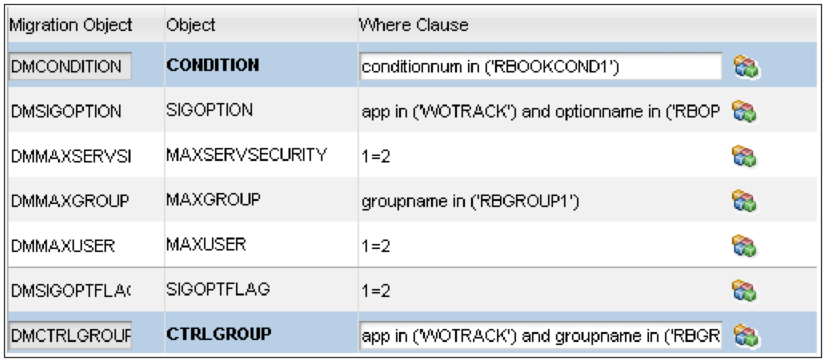
Table 6-11 RBUIPKG01 migration package values

Field	Value
Package Definition Name	RBUIPKG01
Description	IBM Redbooks UI Security Pkg 01
Type	SNAPSHOT
Processing Action	AddChange

Under the Migration Groups tab, we added the RBSECURITY01 migration group.

Defining SQL criteria

Figure 6-14 shows the queries that we used to filter the data that was created in this example.



Migration Object	Object	Where Clause
DMCONDITION	CONDITION	conditionnum in ('RBOOKCOND1')
DMSIGOPTION	SIGOPTION	app in ('WOTRACK') and optionname in ('RBOP
DMMAXSERVSI	MAXSERVSECURITY	1=2
DMMAXGROUP	MAXGROUP	groupname in ('RBGROUP1')
DMMAXUSER	MAXUSER	1=2
DMSIGOPTFLA(	SIGOPTFLAG	1=2
DMCTRLGROUF	CTRLGROUP	app in ('WOTRACK') and groupname in ('RBGR

Figure 6-14 Object structure queries

The complete Where clause for the migration object DMSIGOPTION is shown in the following example:

```
app in ('WOTRACK') and optionname in ('RBOPTION1')
```

The complete Where clause for the migration object DMCTRLGROUP is shown in the following example:

```
app in (‘WOTRACK’) and groupname in (‘RBGROUP1’) and optionname in
(‘RBOPTION1’)
```

Tip: Certain queries are defined as 1=2. You do not need to create a migration group for other security migration-related scenarios. You must create only other packages and define separate filters.

If an incorrect SQL Where clause is entered, the Migration Manager application reports the BMXAA7010E error. The SQL Where clause must be corrected before the package definition can be saved.

Be careful when you try to copy and paste the SQL Where clause of the examples. We prefer to enter them manually to avoid any validation errors.

Because we created new options for an existing application in this example, we must filter them to avoid deployment problems. If you create an application or clone an existing application, and you want to migrate all of the options of that application, you do not need to filter them individually. Merely filter them by using the following application name:

```
app in (‘MYNEWAPP1’, ‘MYNEWAPP2’)
```

It filters all of the options for both applications. For more information, see, “Deployment considerations” on page 144.

DMSIGOPTION: For the DMSIGOPTION object structure query, in case you migrate the conditional user interface configuration of a new application, you can filter by the following application name and group names:

```
app in (‘MYNEWAPP1’, ‘MYNEWAPP2’) and groupname in (‘MYGRP1’,
’MYGRP2’)
```

It filters all of the control security for the groups and applications.

Deployment

After you save and approve the package definition, you can proceed with the creation, distribution, and deployment.

Deployment considerations

For this scenario, you must observe the important points that we describe in this section to successfully have your UI security working in the same way as it works in the source environment.

Reference: All of the deployment considerations in “Deployment considerations” on page 129 also apply to this scenario.

New applications

If you create sigoptions for new applications, make sure that the applications exist in the target environment before this migration. It is possible to migrate the sigoptions of a new application in the same package as the application. For more information, see Chapter 8, “Migrating applications and Start Centers” on page 205.

New users

Although we did not add new users to this scenario, it is supported by using, for example, the approach that is described in 6.1, “Migrating a new security group” on page 122.

6.2.3 Solution B: Migration collection

For this section, because you already created the configurations for solution A, you can browse to the respective applications and add the configurations to a new collection for the Conditional expressions. Ensure that you enabled the Add to Migration Collection button in the relevant applications.

Define collection

By using the system’s navigation menu, you click **Go To** → **System Configuration** → **Migration** → **Migration Collections** and create the collection. Then, proceed to the following applications:

- Conditional Expression Manager
- Security groups
- Application Designer

In each of these applications, browse to the various indicated configurations as shown in 6.2.2, “Solution A: Snapshot” on page 137. When the record is listed on the main tab of the application, click the **Add to Migration Collection** icon and select the collection to which you want to add the configuration.

Validate collection

Now that the collection is defined, you must validate it. Click **Select Action** → **Validate Migration Collection**. After you validate the collection, review the validation configurations. In this scenario, another set of configurations that has to do with signature options, menus, relationships, and other dependent configurations are all added as part of the validation. When you are satisfied with the review, add the validation entries that you need.

Package definition

Click **Select Action** → **Create Package Definition**. From the package definition tab, you can browse to the Migration manager application for deployment by clicking the **Go To Migration Manager** icon, as shown in Figure 6-10 on page 136.

Deployment

After you approve the package definition, you can proceed with the creation, distribution, and deployment.

6.3 Migrating global data restrictions

Data restrictions can be created to a particular set of groups or to the whole system. They are in the object level or the application level. Global data restrictions are restrictions that apply to all of the groups in the system and are in the object level. They are created based on the following example business requirements:

- Canceled or closed work orders must be filtered out from any listing, independently of the user group or application that is used to list the records.
- Deleted users must be filtered out from any listing, independently of the user group or application that is used to list the records.

This scenario describes the migration of a new global data restriction and its Related Data. To perform the tasks that we describe here, you must sign in as an administrative user. For more information about global data restriction in the application level, see Chapter 8, “Migrating applications and Start Centers” on page 205.

6.3.1 Requirements

The requirements for this migration consist of defining a new global data restriction to filter out closed or canceled work orders from any list. The global data restriction uses a new condition, which also is part of this migration requirement.

Global data restrictions and conditions are defined in a development environment and then promoted to the test and production environments.

6.3.2 Solution A: Snapshot

We define an approach so that a new global restriction and its related condition are migrated in two Snapshot migration packages.

The migration of global data restrictions that is described in this scenario implies the migration of the object structure configurations in the first migration package.

The migration of global data restrictions that is described in this scenario implies the migrations of the conditions and security restrictions configurations in the second migration package.

Security restrictions: When associated with a security group, security restrictions are migrated along with the group, as described in 6.1, “Migrating a new security group” on page 122.

Configuration applications

The following sections show the data that was created as an example for this migration scenario.

Conditional Expression Manager

Conditions are SQL expressions or custom Java classes that are used, for example, to limit data access or editability. The restriction that the condition defines is only applied when the conditional expression evaluates to true. Conditions are created and managed by using the Conditional Expression Manager application, which is accessible by selecting **Go To** → **Administration** → **Conditional Expression Manager**.

For this scenario, we created a condition with the values that are shown in Table 6-12.

Table 6-12 RBOOKCOND2

Field	Value
Condition	RBOOKCOND2
Description	IBM Redbooks Condition 2
Type	EXPRESSION
Object Name	WORKORDER
Expression	:status = ‘CLOSE’ or :status = ’CAN’
Always Evaluate	Selected

Security groups

You can create and manage global data restrictions by using the Security Groups application, which is accessed by clicking **Go To** → **Security** → **Security Groups** and then clicking **Select Action** → **Global Data Restriction**.

For this scenario, we created the object data restriction with the values that are shown in Table 6-13.

Table 6-13 Object data restriction

Field	Value
Object	WORKORDER
Type	HIDDEN
Reevaluate	Selected
Condition	RBOOKCOND2

Important: You can use the same condition for separate data restrictions. In this example, when you set the Object field with `WORKORDER`, you are creating this association.

You can associate the same condition with other objects, if the condition can be applied to the other objects.

Object structures

For this scenario, we must create an object structure for global data restrictions. You can create new object structures by using the Object Structure application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Object structures**.

The configurations of the global data restrictions are in the SECURITYRESTRICT table, which is defined as one child in the DMMAXGROUP object structure. In that object structure, there is a relationship between the security group and the data restriction that is based on the group, which is not a requirement of this scenario.

Because there is no predefined object structure in the product that we can use to achieve our migration requirements, we created an object structure with the values that are shown in Table 6-14. All of the other fields feature default values.

Table 6-14 DMSECURITYRESTRICT object structure

Field	**Value**
Object Structure	DMSECURITYRESTRICT
Description	Global Restrictions object structure
Consumed by	MIGRATIONMGR

Under the Source Objects tab, we add the SECURITYRESTRICT object as member of the object structure.

For more information about the DMCONDITION object structure that this scenario needs, see “Object structures” on page 125.

Migration groups

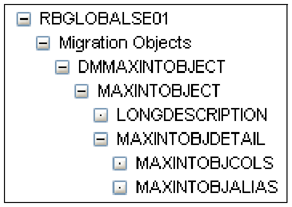
A suitable and efficient approach to migrate the data that we created is to define two new migration groups.

Table 6-15 shows the values of the first migration group. All of the other fields feature default values.

Table 6-15 RBGLOBALSE01 migration group

Field	Value
Migration Group	RBGLOBALSE01
Description	IBM Redbooks Global Secur Obj Structure

This group contains only the DMMAXINTOBJECT migration object. Figure 6-15 shows the migration tree of this group.



```
⊟ RBGLOBALSE01
  ⊟ Migration Objects
    ⊟ DMMAXINTOBJECT
      ⊟ MAXINTOBJECT
        ⊡ LONGDESCRIPTION
        ⊟ MAXINTOBJDETAIL
          ⊡ MAXINTOBJCOLS
          ⊡ MAXINTOBJALIAS
```

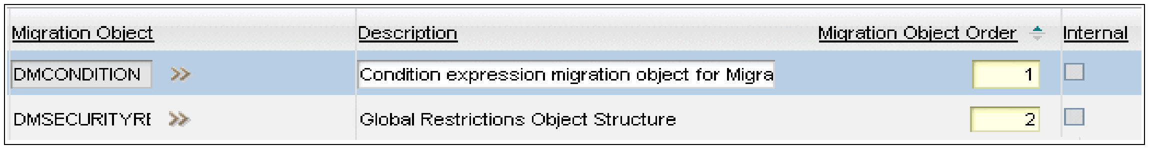
Figure 6-15 RBGLOBALSE01 migration group tree

Table 6-16 shows the values of the second migration group. All of the other fields feature default values.

Table 6-16 RBGLOBALSE02 migration group

Field	Value
Migration Group	RBGLOBALSE02
Description	IBM Redbooks Global Secur Data

This group contains the DMCONDITION and DMSECURITYRESTICT object structures, as shown in Figure 6-16.



Migration Object	Description	Migration Object Order	Internal
DMCONDITION	Condition expression migration object for Migra	1	☐
DMSECURITYRE	Global Restrictions Object Structure	2	☐

Figure 6-16 RBGLOBALSE02 migration group

Figure 6-17 shows the migration tree of this group.

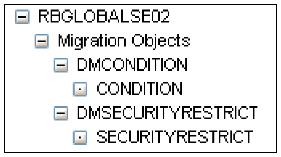

RBGLOBALSE02
Migration Objects
DMCONDITION
CONDITION
DMSECURITYRESTRICT
SECURITYRESTRICT

Figure 6-17 RBGLOBALSE02 migration group tree

Package definitions

We now have the data set and the migration group defined, and the migration process can begin. The first step is to define the package by using the Migration Manager application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Migration Manager**.

Because we are migrating a new object structure, two Snapshot packages are needed to migrate the global data restrictions. The first package migrates the new object structure, and the second package migrates the security restrictions.

Alternative: You can choose to not migrate the new object structure and to create it directly in the target environment. For more information, see , "Deployment considerations" on page 152.

Table 6-17 shows the values of the first package. All of the other fields feature default values.

Table 6-17 RBGLOBALSEC01PKG migration package values

Field	Value
Package Definition Name	RBGLOBALSEC01PKG
Description	IBM Redbooks Global Secur object structure
Type	SNAPSHOT
Processing Action	AddChange

Under the Migration Groups tab, we added the RBGLOBALSE01 migration group.

Table 6-18 on page 151 shows the values of the second package. All of the other fields feature default values.

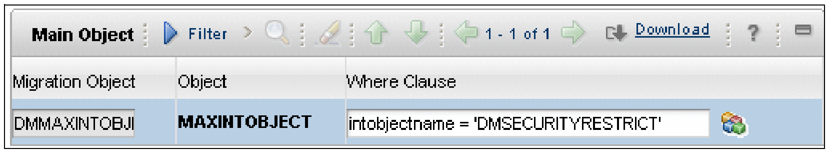
Table 6-18 RBGLOBALSEC02PKG migration package values

Field	Value
Package Definition Name	RBGLOBALSEC02PKG
Description	IBM Redbooks Global Secur Data
Type	SNAPSHOT
Processing Action	AddChange

Under the Migration Groups tab, we added the RBGLOBALSE02 migration group.

Defining SQL criteria

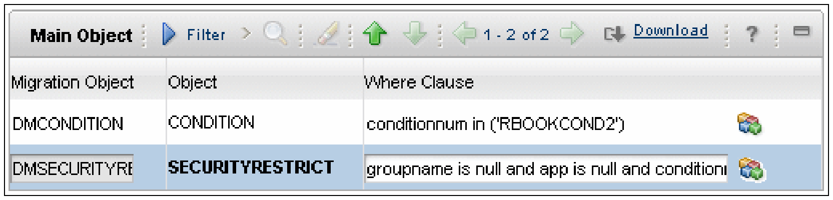
Figure 6-18 shows the queries that we used to filter data for the RBGLOBALSEC01PKG package.



Migration Object	Object	Where Clause
DMMAXINTOBJ	MAXINTOBJECT	intobjectname = 'DMSECURITYRESTRICT'

Figure 6-18 Queries for RBGLOBALSEC01PKG package

Figure 6-19 shows the queries that we used to filter data for the RBGLOBALSEC02PKG package.



Migration Object	Object	Where Clause
DMCONDITION	CONDITION	conditionnum in ('RBOOKCOND2')
DMSECURITYRE	SECURITYRESTRICT	groupname is null and app is null and condition

Figure 6-19 RBGLOBALSEC02PKG object structure query

The following complete query is used to filter the DMSECURITYRESTRICT object structure:

```
groupname is null and app is null and conditionnum in (‘RBOOKCOND2’)
```

Deployment

After we save and approve the two package definitions, we can proceed with the creation, distribution, and deployment. We must migrate the packages separately and in the following order:

- RBGLOBALSEC01PKG
- RBGLOBALSEC02PKG

Deployment considerations

For this scenario, you must observe the important points that we describe in this section for your global security to work successfully in the target environment in the same way that it works in the source environment.

Admin mode

To avoid side effects to the users that are connected to the system, you must turn on the Admin mode in the target environment before the package deployment.

Migrating the new object structure

You must migrate the new object structure that we created to the target environment before the global security migration. During the creation of the packages, the system prompts you about data dictionary migration. You must continue with the process because that prompt is a warning message that is shown every time that a new custom object structure is migrated. It is assumed that no data dictionary changes are pending that you must migrate.

You must deploy the package that contains the new object structure before the package that contains the global security configuration is deployed.

An alternative to the migration of a package with the new object structure is to manually create the object structure in the target environment and then migrate only the second package.

Migrating the global attribute restrictions

We did not describe the migration of global attribute restrictions. You can migrate them in the same way that we migrated the object restrictions because they are stored in the same table. The only difference is the query that we defined filters based on the `RBOOKCOND2` condition. If there is any attribute-level restriction that references that condition, it is migrated, too.

Conditional expression dependencies

Global security restrictions are related with expression conditions. If the conditions are not in the migration package as shown in this scenario, they must be in the target environment before the migration of the global restrictions package.

6.3.3 Solution B: Migration collection

For this section, you must define a new Lookup rule in the Migration Collections application to look up global data restrictions that are defined in the SECURITYRESTRICT table and bring them into the collection. Other required configurations can be added by using any of the available data collection techniques.

Define Lookup Rule

Global data restrictions are defined through a dialog in the Security Groups application. Global data restrictions are not associated with other primary configurations such as security groups. So, a lookup rule is defined to identify and select the data restrictions. Enabling this rule ensures that the identified data restrictions are directly added into the collection.

The following two-step approach is required to define the new lookup rule:

1. Browse to the Domains application and open the DMCOLLLOOKUP domain. Click **New Row** in this domain and add in an ALN Domain value. Set the Value as SECURITYRESTRICT. Provide a description “Global data restrictions table”. Save the changes to the domain.
2. Browse to the Migration Collections application and open the Related Data and Lookup Rules window. Select the **Lookup Rules** tab. Specify a rule name: SECURITYRESTRICT. Provide a description “Global data restrictions records”. Specify SECURITYRESTRICT as the Primary Object. Specify the SQL Where value to be “groupname is NULL”. Enable the rule by selecting the Enabled? option. Click **OK** to save the rule.

Important: A Show Lookup Records button is shown in the Details section of the Lookup rule. This button is provided from a usability perspective: it enables implementers to review the results of SQL criteria they define for a particular lookup rule. For each custom lookup rule, more configuration is required to implement a custom lookup results dialog that shows the wanted attributes of the lookup upon clicking the button. The custom results dialog must be implemented by exporting the Migration Collections presentation (`deplcolls.xml`), editing the custom dialogs that are included in the presentation XML and importing the presentation back into the product environment by using Application Designer. Without this customization, clicking the Show Lookup Records button might result in a BMXAA8122E error message.

Define collection

By using the system’s navigation menu, you click **Go To** → **System Configuration** → **Migration** → **Migration Collections** and create the collection. Proceed to the Conditional Expression Manager application.

In each of these applications, browse to the various configurations that are indicated, as shown in 6.3.2, “Solution A: Snapshot” on page 146. When the record is listed on the main tab of the application, click the **Add to Migration Collection** icon and select the collection to which you want to add the configuration.

Validate collection

Now that the collection is defined, you must validate it. Click **Select Action** → **Validate Migration Collection**. After you validate the collection, review the validation configurations. Because you enabled the SECURITYRESTRICT lookup rule, you should see all available global restrictions in your collection. You can delete any global data restriction entries that you do not need to migrate.

Package definition

Click **Select Action** → **Create Package Definition**. From the package definition tab, you can browse to the Migration manager application for deployment by clicking the **Go To Migration Manager** icon.

6.4 Migrating access definitions and LDAP information

You can perform authentications in the product by using the native authentication of the system or by using the application server security and LDAP.

When you configure your system to authenticate by using application server security that is connected to an LDAP directory server, there is one cron task called Virtual Member Management Synchronization (VMMSYNC) that updates the system with the groups and users that are changed in the directory server.

After you load the groups and users into the system, you can configure the access definitions to them. This scenario is common when you use LDAP-based authentication.

Important: To perform the tasks that we describe here, you must sign in as an administrative user.

6.4.1 Requirements

The requirements of this scenario consist of migrating only the access definitions for the application options.

Access definitions are defined in a development environment and then promoted to the test and production environments.

Important: Security groups and users are not migrated by using the Migration Manager application if LDAP synchronization is set up and application server security is enabled.

This process uses external tools and is not part of the scope of this book.

6.4.2 Solution A: Snapshot

We define an approach so that a set of new access definitions and related conditions are migrated in two Snapshot migration packages.

The migration of access definitions that is described in this scenario implies the migration of the object structure configurations in the first migration package and the migration of the authorizations configurations in the second migration package.

Configuration applications

The following sections show the data that was created as an example for this migration scenario.

Security groups

For this scenario, we use the group that we created in “Security groups” on page 124, which, in this context, means a group that we loaded from the LDAP server directory.

Important: We assume that this group exists in the target environment and has no access definitions. For more information, see , “Deployment considerations” on page 159.

Under the Applications tab, the group received access to all of the options of the People application.

Object structures

For this scenario, we must create an object structure for the access definitions. You can create object structures by using the Object Structure application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Object structures.**

The data access configurations are in the table APPLICATIONAUTH, which is defined as one child in the object structure DMMAXGROUP. In that object structure, there is a relationship between the security group and the access definitions that is based on the group, which is not a requirement of this scenario.

Because there is no predefined object structure in the product that we can use to achieve our migration requirements, we created an object structure with the values that are shown in Table 6-19. All of the other fields feature default values.

Table 6-19 DMAPPLICATIONAUTH object structure

Field	Value
Object Structure	DMAPPLICATIONAUTH
Description	Authorizations
Consumed by	MIGRATIONMGR

Under the Source Objects tab, we added the APPLICATIONAUTH object.

Migration groups

A suitable and efficient approach to migrate the data we that created is to define two new migration groups.

Table 6-20 shows the values of the first migration group. All of the other fields feature default values.

Table 6-20 RBAUTHORIZATIONS01 migration group

Field	Value
Migration Group	RBAUTHORIZATIONS01
Description	IBM Redbooks Authorizations Obj Structure

This group contains only the object structure DMMAXINTOBJECT.

Figure 6-20 shows the migration tree of this group.

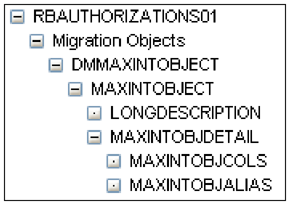


Figure 6-20 RBAUTHORIZATIONS01 migration group tree

Table 6-21 shows the values of the second migration group. All other fields feature default values.

Table 6-21 RBAUTHORIZATIONS02 migration group

Field	Value
Migration Group	RBAUTHORIZATIONS02
Description	IBM Redbooks Access Definitions Data

This group contains the object structure DMAPPLICATIONAUTH. Figure 6-21 shows the migration tree of this group.

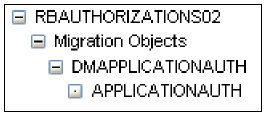


Figure 6-21 RBAUTHORIZATIONS02 migration group tree

Package definitions

The data set and migration group is now defined and the migration process can begin. The first step is to define the package by using the Migration Manager application, which is accessed by clicking **Go To** → **System Configuration** → **Migration** → **Migration Manager**.

Two Snapshot packages are required to migrate the data access. The first package migrates the new object structure that we created, and the second package migrates access definitions and conditions.

Alternative: You can choose to not migrate the new object structure and instead to create it directly in the target environment. For more information, see, “Deployment considerations” on page 159.

Table 6-22 shows the values of the first package. All other fields feature default values.

Table 6-22 RBAUTH01PKG migration package values

Field	Value
Package Definition Name	RBAUTH01PKG
Description	IBM Redbooks Auth object structure
Type	SNAPSHOT
Processing Action	AddChange

Under the Migration Groups tab, we added the migration group RBAUTHORIZATIONS01.

Table 6-23 shows the values of the second package. All other fields feature default values.

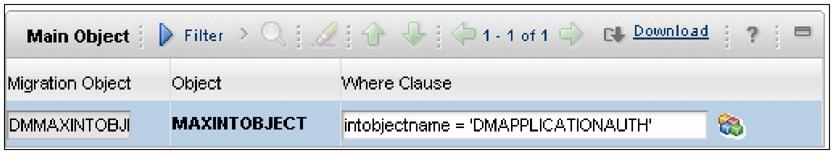
Table 6-23 RBAUTH02PKG migration package values

Field	Value
Package Definition Name	RBAUTH02PKG
Description	IBM Redbooks Auth data
Type	SNAPSHOT
Processing Action	AddChange

Under the Migration Groups tab, we added the migration group RBAUTHORIZATIONS02.

Defining SQL criteria

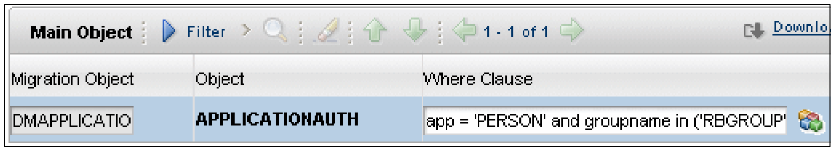
Figure 6-22 shows the queries that we used to filter the data for the package RBAUTH01PKG.



Migration Object	Object	Where Clause
DMMAXINTOBJ	MAXINTOBJECT	intobjectname = 'DMAPPLICATIONAUTH'

Figure 6-22 Queries for package RBAUTH01PKG

Figure 6-23 shows the queries that we used to filter data for package RBAUTH02PKG.



Migration Object	Object	Where Clause
DMAPPLICATIO	APPLICATIONAUTH	app = 'PERSON' and groupname in ('RBGROUP'

Figure 6-23 Queries for package RBAUTH02PKG

The following query is used to filter the DMAPPLICATIONAUTH migration object:

```
app = ‘PERSON’ and groupname in (‘RBGROUP’)
```

Deployment

After you saved and approved the two package definitions, you can proceed with the creation, distribution, and deployment. You must migrate the packages separately and in the following order:

- RBAUTH01PKG
- RBAUTH02PKG

Deployment considerations

For this scenario, you must observe the important points that we describe in this section to successfully have your global security work in the same way as it works in the source environment.

Admin mode

To avoid side effects to users that are connected to the system, you must enable the Admin mode in the target environment before the package deployment.

Migrating the new object structure

You must migrate the new object structure that we created to the target environment before the access definitions. During the creation of the packages, the system prompts you about the data dictionary migration. Continue with the process because this warning message is displayed every time that a new custom object structure is migrated. It is assumed that there are no pending data dictionary changes that you must migrate.

The deployment of the package that contains the new object structure must be done before the deployment of the package that contains the data access configuration.

An alternative to migrating the package with the new object structure is to create the object structure manually in the target environment.

Metadata assumptions

We assume that your product configuration did not modify the primary key of the APPLICATIONAUTH table or extend its object.

Application and security group dependencies

The application definitions, security groups, and application options for which you defined access must exist in the target environment.

Conditional expression dependencies

Access definitions are related with expression conditions. If the conditions are not in the migration package, they must be in the target environment before the migration of the access definitions package.

Considerations about migrating access definitions

This scenario covered the first migration of access definitions, which means that the data that you are migrating does not exist in the target environment.

If you must make changes in the source environment and perform a new migration, you can use almost the same approach that we presented in this section, but consider a Replace processing action or use a Change package. For more information, see Chapter 15, “Common topics” on page 351.

6.4.3 Solution B: Migration collection

A custom lookup rule is required to collect the wanted application access definitions. This rule is defined in the same two-step approach that is described in 6.3.3, “Solution B: Migration collection” on page 153. The rule targets the APPLICATIONAUTH business object. The SQL criteria that is associated with the lookup rule must be defined to match the criteria that is defined in Solution A. The remainders of the steps are similar to Solution B that is outlined in 6.3, “Migrating global data restrictions” on page 145.

6.5 Considerations of security migrations

This chapter presented several scenarios for security migration. These scenarios are presented individually and do not depend on each other. However, you can combine or extend them to achieve new migration scenarios.

6.6 Considerations of the use of collections versus snapshots or change packages

The descriptions in the previous four scenarios in this chapter highlight some significant considerations in the use of collections versus snapshots or change packages. You first must consider that in two of the four scenarios, we needed to create new object structures (because the existing object structures contained too much metadata thus hindering performance or there were no ready for use object structures to migrate these objects) to migrate the configurations.

In so, doing we also needed to migrate the DMMAXINTOBJECT object structure to create the object structure in the target environment. This process is time consuming and if forgotten, can lead to costly troubleshooting.

Collections solve this situation because you can migrate configurations independent from predefined migration groups. By using data validation, you can select those configurations that are truly necessary and leave behind redundant and preexisting metadata that does not require migration.

7

Migrating escalations that include automation scripts

Escalations are server-side functions that automate actions and notifications in any Tivoli's process automation engine-based product. Escalations are used extensively by clients and practitioners to automate many aspects of the business that are supported by Tivoli's process automation engine-based products. For example, an escalation monitors service desk compliance with the commitments of a service-level agreement (SLA). If the response time for a high priority service request (SR) is approaching the defined SLA threshold that is defined in a commitment, the escalation fires actions and notifications. The actions might increase the severity of the SR. The notifications might alert a service desk manager of approaching noncompliance.

This chapter describes the best practices in migrating escalations and includes the following sections:

- Requirements
- Solution
- Configuration applications
- Object structures
- Migration groups
- Package definitions
- Deployment
- Migrating escalations with collections
- Requirements
- Solution
- Configuration applications
- Creating a migration collection
- Using Related rules to find the Action launch point
- Deployment

7.1 Requirements

Escalations are defined in a development environment and then promoted to test production environments. A repeatable process is required to reliably (without any errors or failures) and accurately (without missing any required content) migrate escalation configurations and any related configurations.

7.2 Solution

We define an approach so that an escalation configuration and all related configurations are migrated in a single Snapshot migration package.

From a content perspective, the migration of an escalation primarily implies the migration of the following configurations in order:

- Actions
- Action groups
- Person
- Person group
- Role
- Communication template
- Escalations

7.3 Configuration applications

This section describes the applications that support an escalation configuration. The section also describes dependencies among these applications that drive the migration of escalations.

7.3.1 Actions

You can access actions and action groups from the appropriate product Start Center by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Actions**.

7.3.2 People

Whether a role or a communication template is associated with a person record, creation and management of the person record is done by using the People application. You can access person records from the appropriate product Start Center by clicking **Go To** → **Administration** → **Resources** → **People**.

7.3.3 Person groups

The creation and management of the person group record is done by using the Person Groups application. You can access person groups from the appropriate product Start Center by clicking **Go To** → **Administration** → **Resources** → **Person Groups**.

The configurations and governing applications that were identified thus far constitute the primary dependencies for an escalation.

7.3.4 Roles

When a communication template is associated with a role, the role is defined by using the Roles application. You can access roles from the appropriate product Start Center by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Roles**.

A role can be built from a person or person group record.

7.3.5 Communication templates

Notifications are managed by using the Communication Templates application. You can access communication templates from the appropriate product Start Center by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Communication Templates**.

A communication template, in turn, can define one or more recipients in the form of email address, role, person, or person group records.

7.3.6 Escalations

Escalations are created and managed by using the Escalations application. You can access escalations from the appropriate product Start Center by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Escalations**.

An escalation starts one or more actions and notifications. Therefore, every escalation is associated with a corresponding set of actions or notifications.

7.3.7 Other applications

In addition to these configurations, an escalation depends on another set of configurations. The following configurations are secondary in the sense that they might be delivered with the product or might be previously migrated:

- Cron tasks

  An escalation always requires a cron task instance to run in a periodic manner. Cron tasks are defined with the Cron Task Setup application. All escalation cron task instances are derived from the ESCALATION cron task that is delivered with the product. When an escalation is activated, a cron task instance is created if a cron task instance does not exist. The dynamic creation of a cron task instance implies that there are alternative approaches to migrating the cron task instance that is associated with an escalation.

- Organization and site

  An escalation might specify an association to an organization or site. Organizations and sites are defined with the Organizations application. For this migration scenario, it is assumed that the organizations and sites are already migrated.

- Business object

  An escalation always specifies an object that is the Target of the escalation actions and notifications. In fact, roles, communication templates, and escalations all depend on the existence of a business object. A business object is defined with the Database Configuration application. Most roles, communication templates, and escalations are defined for business objects that are delivered with the product. For this migration scenario, it is assumed that the business object definitions are already migrated.

We established that an escalation is built not in isolation but with configurations that are managed through other applications. Escalation migration must take into account all of these related configurations. Figure 7-1 shows these dependencies.

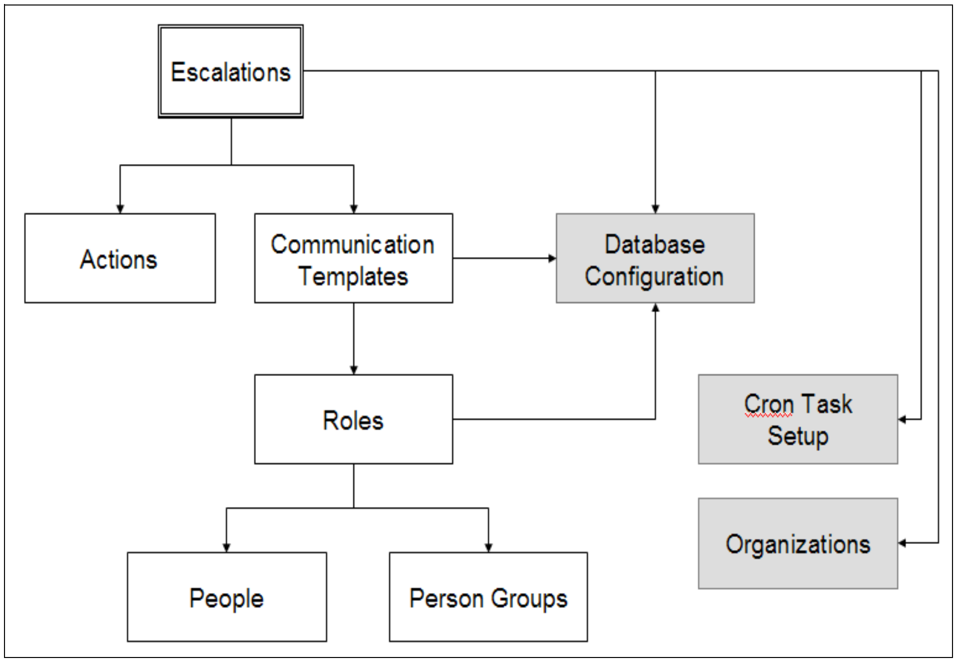
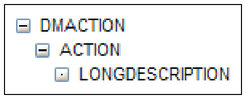


Figure 7-1 Escalations and related applications

7.4 Object structures

The following object structures are delivered with the product.

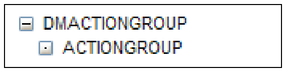
The DMACTION object structure, as part of the business process management (BPM) migration group, supports action group migration. Figure 7-2 shows the contents of the DMACTION object structure.



```
⊟ DMACTION
  ⊟ ACTION
    ⊡ LONGDESCRIPTION
```

Figure 7-2 DMACTION object structure

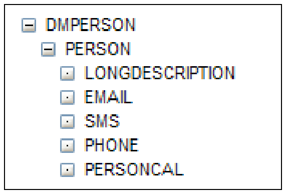
The DMACTIONGROUP object structure, as part of the BPM migration group, supports action migration. Figure 7-3 shows the contents of the DMACTIONGROUP object structure.



```
⊟ DMACTIONGROUP
  ⊡ ACTIONGROUP
```

Figure 7-3 DMACTIONGROUP object structure

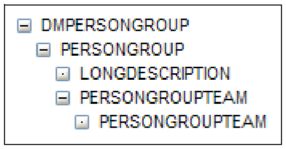
The DMPERSON object structure, as part of the RESOURCES migration group, supports person configuration migration. Figure 7-4 shows the contents of the DMPERSON object structure.



```
⊟ DMPERSON
  ⊟ PERSON
    ⊡ LONGDESCRIPTION
    ⊡ EMAIL
    ⊡ SMS
    ⊡ PHONE
    ⊡ PERSONCAL
```

Figure 7-4 DMPERSON object structure

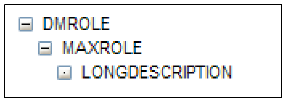
The DMPERSONGROUP object structure, as part of the RESOURCES migration group, supports person group migration. Figure 7-5 shows the contents of the DMPERSONGROUP object structure.



- DMPERSONGROUP
  - PERSONGROUP
    - LONGDESCRIPTION
    - PERSONGROUPTEAM
      - PERSONGROUPTEAM

Figure 7-5 DMPERSONGROUP object structure

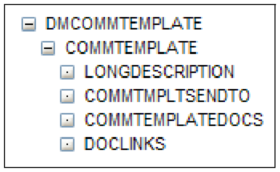
The DMROLE object structure, as part of the BPM migration group, supports role migration. Figure 7-6 shows the contents of the DMROLE object structure.



- DMROLE
  - MAXROLE
    - LONGDESCRIPTION

Figure 7-6 DMROLE object structure

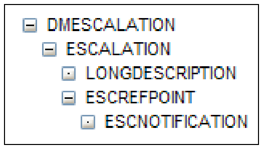
The DMCOMMTEMPLATE object structure, as part of the BPM migration group, supports the communication template migration. Figure 7-7 shows the contents of the DMCOMMTEMPLATE object structure.



- DMCOMMTEMPLATE
  - COMMTEMPLATE
    - LONGDESCRIPTION
    - COMMTMPLTSENDTO
    - COMMTEMPLATEDOCS
    - DOCLINKS

Figure 7-7 DMCOMMTEMPLATE object structure

The DMESCALATION object structure, as part of the BPM migration group, supports the escalation migration. Figure 7-8 shows the contents of the DMESCALATION object structure.



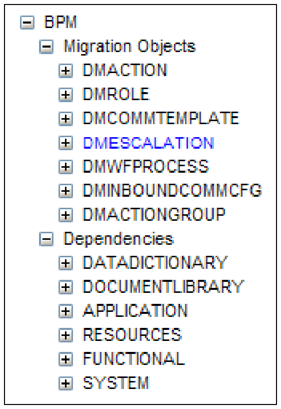
- DMESCALATION
  - ESCALATION
    - LONGDESCRIPTION
    - ESCREFPOINT
      - ESCNOTIFICATION

Figure 7-8 DMESCALATION object structure

7.5 Migration groups

The object structures that support escalation migration belong to two migration groups: BPM and RESOURCES.

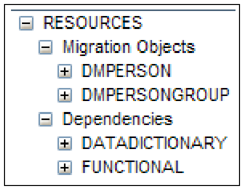
Figure 7-9 shows the BPM migration group and its dependencies as delivered with the product.



```
⊟ BPM
  ⊟ Migration Objects
    ⊞ DMACTION
    ⊞ DMROLE
    ⊞ DMCOMMTEMPLATE
    ⊞ DMESCALATION
    ⊞ DMWFPROCESS
    ⊞ DMINBOUNDCOMMCFG
    ⊞ DMACTIONGROUP
  ⊟ Dependencies
    ⊞ DATADICTIONARY
    ⊞ DOCUMENTLIBRARY
    ⊞ APPLICATION
    ⊞ RESOURCES
    ⊞ FUNCTIONAL
    ⊞ SYSTEM
```

Figure 7-9 BPM migration group

Figure 7-10 shows the RESOURCES migration group and its dependencies as delivered with the product.



```
⊟ RESOURCES
  ⊟ Migration Objects
    ⊞ DMPERSON
    ⊞ DMPERSONGROUP
  ⊟ Dependencies
    ⊞ DATADICTIONARY
    ⊞ FUNCTIONAL
```

Figure 7-10 RESOURCES migration group

The use of the BPM and RESOURCES migration groups is inefficient for the following reasons:

- Only a subset of the configurations is required to be migrated.
- Configurations pertaining to the dependencies that are listed with the two migration groups might already be migrated.

Figure 7-11 shows the object structures and migration groups that pertain to BPM that are not required for the escalation migration scenario.

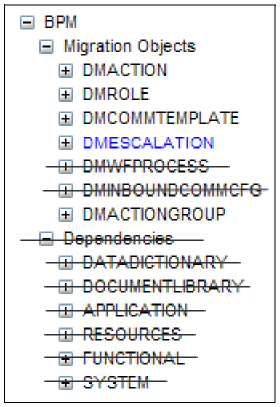

BPM
- Migration Objects
  - DMACTION
  - DMROLE
  - DMCOMMTEMPLATE
  - DMESCALATION
  - ~~DMWFPROCESS~~
  - ~~DMINBOUNDCOMMCFG~~
  - DMACTIONGROUP
- ~~Dependencies~~
  - ~~DATADICTIONARY~~
  - ~~DOCUMENTLIBRARY~~
  - ~~APPLICATION~~
  - ~~RESOURCES~~
  - ~~FUNCTIONAL~~
  - ~~SYSTEM~~

Figure 7-11 Discarded BPM object structures and dependencies

Important The strike-through lines that are shown in Figure 7-11 are only for explanatory purposes. The product does not provide a capability to remove object structures and migration groups in a strike-through manner.

Figure 7-12 shows the object structures and migration groups that pertain to RESOURCES that are not required for this migration scenario.

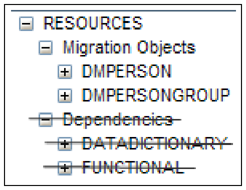
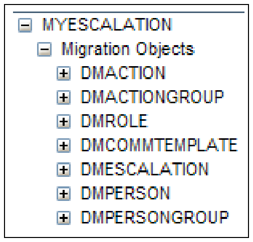

RESOURCES
- Migration Objects
  - DMPERSON
  - DMPERSONGROUP
- ~~Dependencies~~
  - ~~DATADICTIONARY~~
  - ~~FUNCTIONAL~~

Figure 7-12 Discarded RESOURCES dependencies

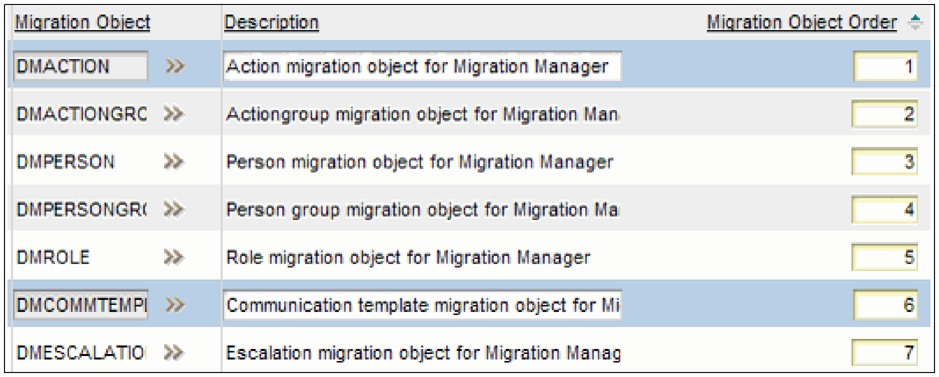
For the escalation migration scenario, create a migration group that contains only the required object structures. The new migration group can be easily constructed by duplicating the existing BPM migration group and modifying its contents. Figure 7-13 shows the new MYESCALATION migration group and its contents.



- MYESCALATION
  - Migration Objects
    - DMACTION
    - DMACTIONGROUP
    - DMROLE
    - DMCOMMTEMPLATE
    - DMESCALATION
    - DMPERSON
    - DMPERSONGROUP

Figure 7-13 Object structures and migration group that supports escalation migration

Figure 7-14 shows the ordering of the object structures within this new migration group.



Migration Object	Description	Migration Object Order
DMACTION	Action migration object for Migration Manager	1
DMACTIONGRC	Actiongroup migration object for Migration Man	2
DMPERSON	Person migration object for Migration Manager	3
DMPERSONGR(	Person group migration object for Migration Ma	4
DMROLE	Role migration object for Migration Manager	5
DMCOMMTEMPI	Communication template migration object for Mi	6
DMESCALATIO	Escalation migration object for Migration Manag	7

Figure 7-14 Object structure order

Important: Figure 7-14 is used to show the object structures and their ordering in the new migration group for explanatory purposes only. The product user interface might differ.

7.6 Package definitions

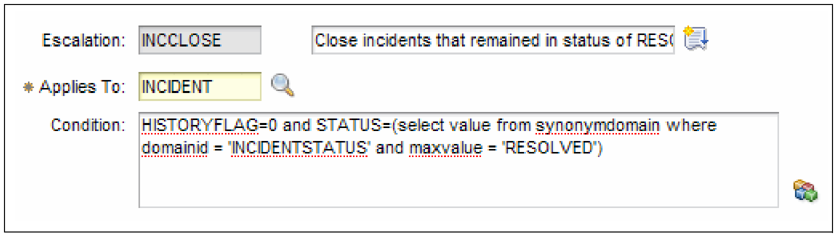
Now that the content is identified and organized, you can begin the migration. As a first step, a Snapshot migration package is defined. The definition includes the created migration group. SQL criteria are associated with the appropriate object structures that are members of the new migration group.

Defaults: Accept all of the default values that the Migration Manager application provides for the package definition.

The following section uses sample escalations that are delivered with the product to show the process that is used to set up SQL criteria.

7.6.1 Defining SQL criteria for an escalation with an action

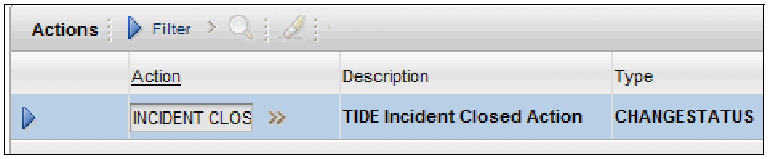
The sample escalation INCCLOSE is used for this description. This escalation is intended to close Incident records in the product that remain in the resolved state for more than 10 days. Figure 7-15 shows the escalation header for the INCCLOSE escalation.



Escalation:	INCCLOSE	Close incidents that remained in status of RES(
* Applies To:	INCIDENT	
Condition:	HISTORYFLAG=0 and STATUS=(select value from synonymdomain where domainid = 'INCIDENTSTATUS' and maxvalue = 'RESOLVED')	

Figure 7-15 INCCLOSE sample escalation

The INCCLOSE escalation features a single escalation point that runs an action, but no associated notifications. Figure 7-16 shows the INCIDENT CLOSED action that is associated with the escalation.



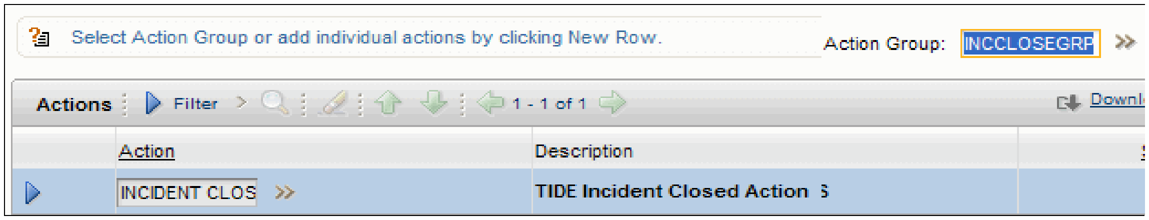
Action	Description	Type
INCIDENT CLOS	TIDE Incident Closed Action	CHANGESTATUS

Figure 7-16 INCIDENT CLOSE action

This action is part of an *action group*. An action group enables multiple actions to be run in sequence when an escalation runs.

Escalation: An escalation is always associated with an action group that, in turn, contains one or more individual actions.

Figure 7-17 shows the INCCLOSEGRP action group of which the INCIDENT CLOSED action is a member.



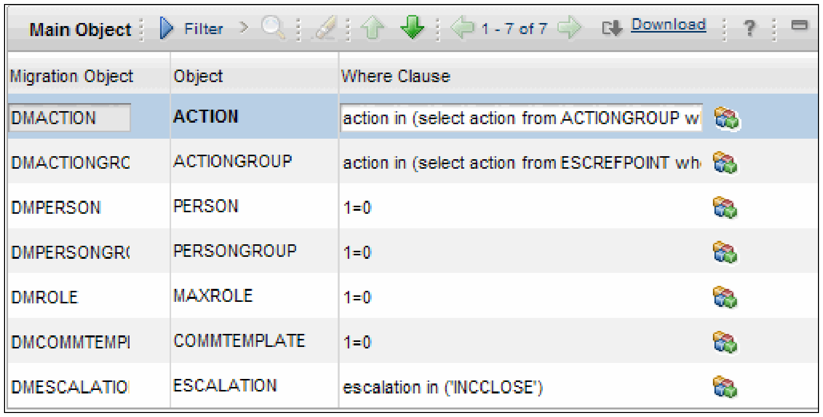
Select Action Group or add individual actions by clicking New Row.

Action Group: INCCLOSEGRP

Action	Description
INCIDENT CLOS	TIDE Incident Closed Action

Figure 7-17 INCCLOSEGRP action group and INCIDENT CLOSED member action

The migration of the INCCLOSE escalation includes the migration of the associated action INCIDENT CLOSED and action group INCCLOSEGRP. Thus, SQL criteria can be set up for the object structures that are part of the migration package. Figure 7-18 shows the specific SQL criteria setup for the escalation, action, and action group object structures.



Migration Object	Object	Where Clause
DMACTION	ACTION	action in (select action from ACTIONGROUP w
DMACTIONGRC	ACTIONGROUP	action in (select action from ESCREFPOINT wh
DMPERSON	PERSON	1=0
DMPERSONGR(	PERSONGROUP	1=0
DMROLE	MAXROLE	1=0
DMCOMMTEMPI	COMMTEMPLATE	1=0
DMESCALATIO	ESCALATION	escalation in ('INCCLOSE')

Figure 7-18 SQL criteria to migrate INCCLOSE escalation

The complete SQL Where clause for the DMACTION object structure is shown in the following example:

```
action in (select action from ACTIONGROUP where action in (select
action from ESCREFPOINT where escalation in     ('INCCLOSE') ) ) or
action in (select member from actiongroup where action in (select
action from escrefpoint where escalation in ('INCCLOSE')))
```

Important: The SQL Where clause for the DMACTION object structure is designed to collect the action groups and individual actions that are required for the escalation. The entries for action groups and individual actions are in the same ACTION table in the underlying production database.

The complete SQL Where clause for the DMACTIONGROUP object structure is shown in the following example:

```
action in (select action from ESCREFPOINT where escalation in
('INCCLOSE') )
```

The complete SQL Where clause for the DMESCALATION object structure is shown in the following example:

```
escalation in ('INCCLOSE')
```

Because no other related configuration is required to be collected for the migration of the INCCLOSE escalation, the other object structures in the migration package specify the SQL Where clause to be set to the following parameter:

```
1=0
```

This SQL Where clause results in no configuration records are picked up for the DMPERSON, DMPERSONGROUP, DMROLE, and DMCOMMTEMPLATE object structures.

Important: Leaving the Where clause field empty implies that all of the configuration records of the primary business object of the object structure must be collected for migration. This result might not be the wanted result. If no configuration record is to be extracted for an object structure, specify the SQL Where clause 1=0.

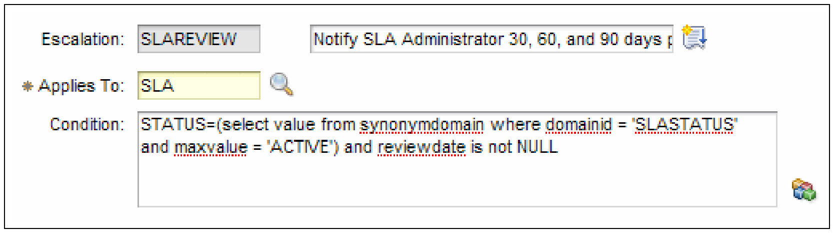
The SQL Where clauses are constructed in a manner so that only the name of the specific escalation must be specified. If a separate escalation is migrated, the same migration package definition can be reused by replacing the INCCLOSE escalation identifier with the identifier of the wanted escalation.

Tip: When the various object structures in a migration group are related to each other, specify the SQL clauses for these object structures in a manner so that they reference the primary object structure. Knowledge of the data models for the various applications is necessary to develop the SQL clauses.

Save the package definition. The remainder of the migration steps can now be performed.

7.6.2 Defining SQL criteria for an escalation with a notification

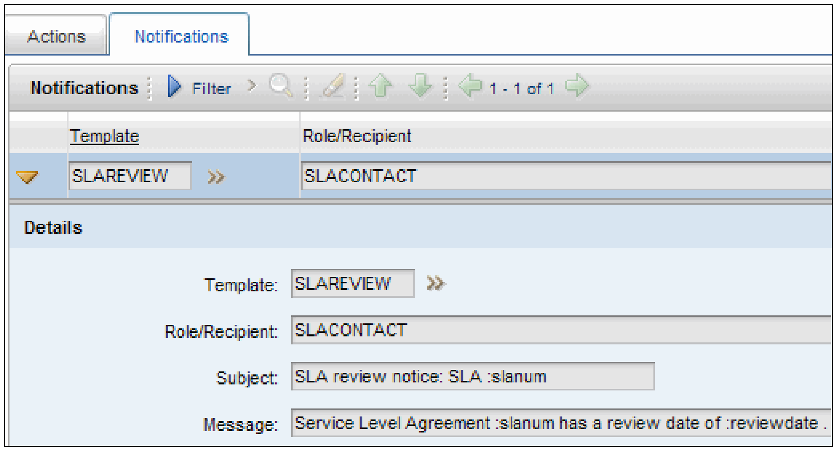
The sample escalation SLAREVIEW is used for this description. This escalation is intended to proactively send SLA review notifications to the designated individual in charge of reviewing an SLA before its expiration. Figure 7-19 shows the escalation header for the SLAREVIEW escalation.



Escalation:	SLAREVIEW — Notify SLA Administrator 30, 60, and 90 days p
* Applies To:	SLA
Condition:	STATUS=(select value from synonymdomain where domainid = 'SLASTATUS' and maxvalue = 'ACTIVE') and reviewdate is not NULL

Figure 7-19 SLAREVIEW sample escalation

The SLAREVIEW escalation has three escalation points, each of which sends a notification, but no associated actions. All three escalation points send the same notification; however, each notification is sent out at a separate time. Figure 7-20 shows the SLAREVIEW communication template that is associated with the escalation.



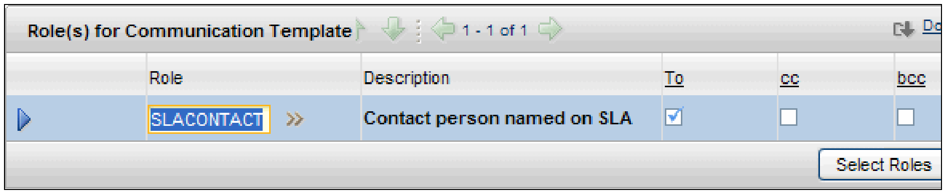
Template	Role/Recipient
SLAREVIEW	SLACONTACT

Details

Template:	SLAREVIEW
Role/Recipient:	SLACONTACT
Subject:	SLA review notice: SLA :slanum
Message:	Service Level Agreement :slanum has a review date of :reviewdate .

Figure 7-20 SLAREVIEW communication template

Use the Detail menu icon to the right of the Template field to open the SLAREVIEW communication template with the Communication Templates application. The Communication Template tab displays the content of the notification, such as the Subject and Message information. The Recipients tab lists various recipient types. Opening the Roles section lists the roles to which this notification is sent. Figure 7-21 shows the Roles section.

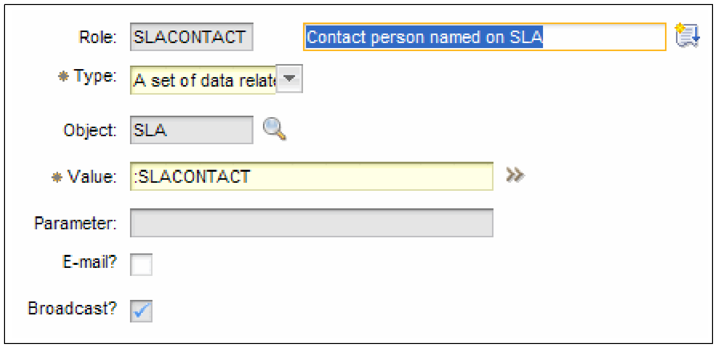


Role(s) for Communication Template 1 - 1 of 1

Role	Description	To	cc	bcc
SLACONTACT	Contact person named on SLA	✓	☐	☐

Figure 7-21 SLACONTACT role for SLAREVIEW communication template

Use the Detail menu icon to the right of the Role field to open the SLACONTACT role with the Roles application. The Role tab displays the content of the role. This role does not depend on a person or person group configuration. Figure 7-22 shows the SLACONTACT role.



Role:	SLACONTACT Contact person named on SLA
* Type:	A set of data relat
Object:	SLA
* Value:	:SLACONTACT
Parameter:	
E-mail?	☐
Broadcast?	✓

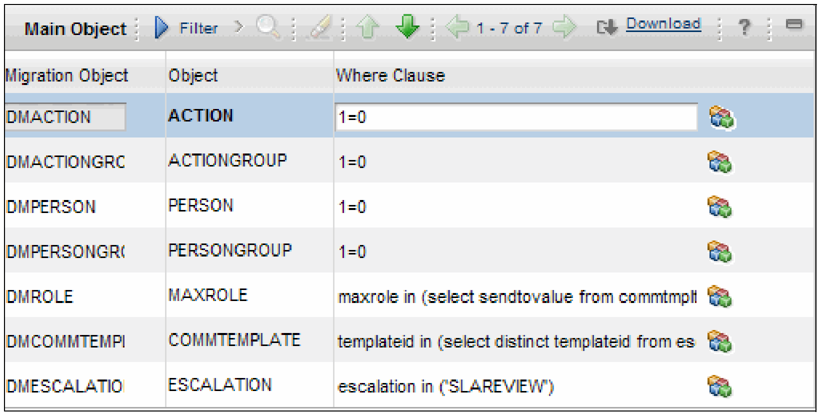
Figure 7-22 SLACONTACT role details

The migration of the SLAREVIEW escalation includes the migration of the associated communication template SLAREVIEW and the role SLACONTACT.

The SLAREVIEW escalation is migrated with the same package definition that was defined earlier. There is no need to create a separate package definition. To reuse the existing package definition, alter the SQL criteria that are associated with the various object structures in the package definition.

Alternatives: Before the SLAREVIEW escalation is migrated, a physical package must be created before the SQL criteria is altered to ensure that the configurations that pertain to the INCCLOSE escalation are extracted as wanted. If a physical package is not yet created and the SQL criteria are altered, it is not possible to extract the configurations that pertain to the INCCLOSE escalation. Alternatively, SQL criteria can be combined to collect the configurations that pertain to INCCLOSE and SLAREVIEW escalations. The choice is driven by implementation requirements.

Thus, SQL criteria can be set up for the object structures that are part of the migration package. Figure 7-23 shows the specific SQL criteria setup for the escalation, action, and action group object structures.



Migration Object	Object	Where Clause
DMACTION	ACTION	1=0
DMACTIONGRC	ACTIONGROUP	1=0
DMPERSON	PERSON	1=0
DMPERSONGR(	PERSONGROUP	1=0
DMROLE	MAXROLE	maxrole in (select sendtovalue from commtmplt
DMCOMMTEMPI	COMMTEMPLATE	templateid in (select distinct templateid from es
DMESCALATIO	ESCALATION	escalation in ('SLAREVIEW')

Figure 7-23 SQL criteria to migrate SLAREVIEW escalation

The complete SQL Where clause for the DMROLE object structure is shown in the following example:

```
maxrole in (select sendtovalue from commtmpltsendto where templateid in
(select distinct templateid from escnotification where escalation in
('SLAREVIEW')))
```

The complete SQL Where clause for the DMCOMMTEMPLATE object structure is shown in the following example:

```
templateid in (select distinct templateid from escnotification where
escalation in ('SLAREVIEW'))
```

The complete SQL Where clause for the DMESCALATION object structure is shown in the following example:

```
escalation in ('SLAREVIEW')
```

Tip: If an incorrect SQL Where clause is entered, the Migration Manager reports the BMXAA7010E error. You must correct the SQL Where clause before the package definition can be saved.

Save the package definition. The remainder of the migration steps can now be performed.

7.7 Deployment

Create, distribute, and deploy the package. The escalation configurations and their related configurations can be deployed into a live production environment. We describe the deployment considerations in this section.

7.7.1 Escalations and cron tasks

Every escalation runs periodically, which uses an underlying cron task instance. All escalation cron task instances are built from the ESCALATION cron task that ships with the product.

When an escalation is activated for the first time, the corresponding cron task instance also is created and linked to the ESCALATION cron task. The cron task instance name includes the name of the corresponding escalation and is preceded by a three letter prefix of ESC. A reference to the cron task instance name is retained in the INSTANCENAME column of the ESCALATION table. For example, the SLAREVIEW escalation carries a reference to the ESCSLAREVIEW cron task instance. After an escalation is deactivated, it continues to retain the reference to the previously created cron task instance.

The following approaches can be used to resolve the dependency of an escalation on a cron task instance:

- Use the Escalation application to create the cron task instance.
- Migrate the cron task and its instances to support an escalation.

Use the Escalation application to create a cron task instance

This approach uses the Escalation application's ability to create the required cron task that supports periodic escalation execution. This approach takes effect when the escalation is migrated for the first time, which results in the insertion of various configuration records in the Target production environment.

Complete the following steps to use the Escalation application to create a cron task instance:

1. Start the INSTANCENAME attribute from the DMESCALATION object structure in the Source and Target production environments by completing the following steps:
   a. Open the Object Structures application.
   b. Find the DMESCALATION object structure in the List tab.
   c. Open the DMESCALATION object structure in the Object Structure tab.
   d. From the Select Action menu, run the Exclude/Include Fields action.
   e. In the Exclude/Include Fields window, ensure that the ESCALATION object is selected in the Source Objects for ESCALATION table window.
   f. In the Persistent fields subtab, scroll until you locate the INSTANCE Field.
   g. Select **Exclude**.
   h. Click **OK** to save the changes and dismiss the Exclude/Include Fields window.
2. Deactivate the wanted escalations in the Source production environment.
3. Define a migration package and associate the appropriate SQL criteria to the object structures in the migration group.
4. Approve, activate, and create the migration package.
5. Distribute the migration package to the Target production environment.
6. Deploy the migration package.
7. After the deployment is complete, open the migrated escalations in the Escalations application and activate each of them.

Upon activation, the Escalation application determines a cron task instance is required and creates the instance and links it back into the escalation. The escalation is now in effect in the Target production environment.

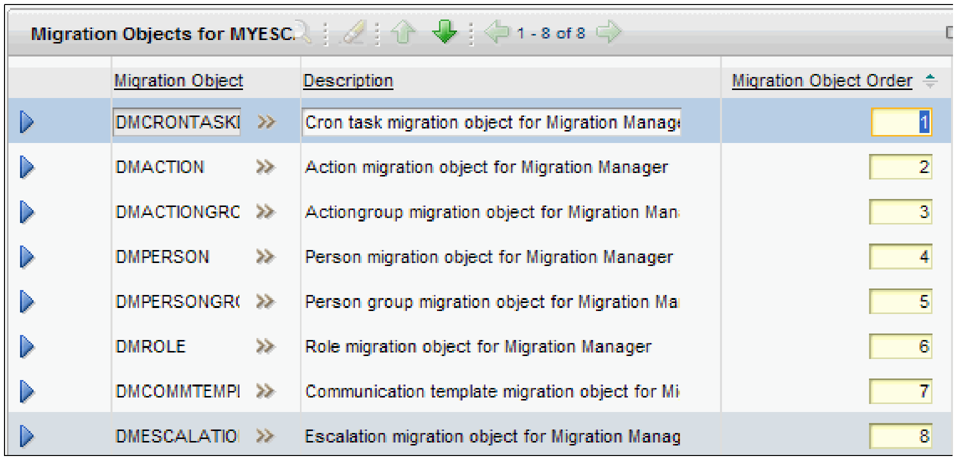
Updating an escalation: This approach is also useful for a migration scenario where the escalation is updated rather than inserted. For an update migration scenario, the cron task instance is already associated with the escalation and this association is left intact during migration.

Migrating escalations with escalation cron task instances

This approach collects all of the cron task instances that are associated with the ESCALATION cron task and migrates them along with the escalations themselves. This approach requires you to perform more steps that result in a longer migration effort.

Complete the following steps to migrate escalations with escalation cron task instances:

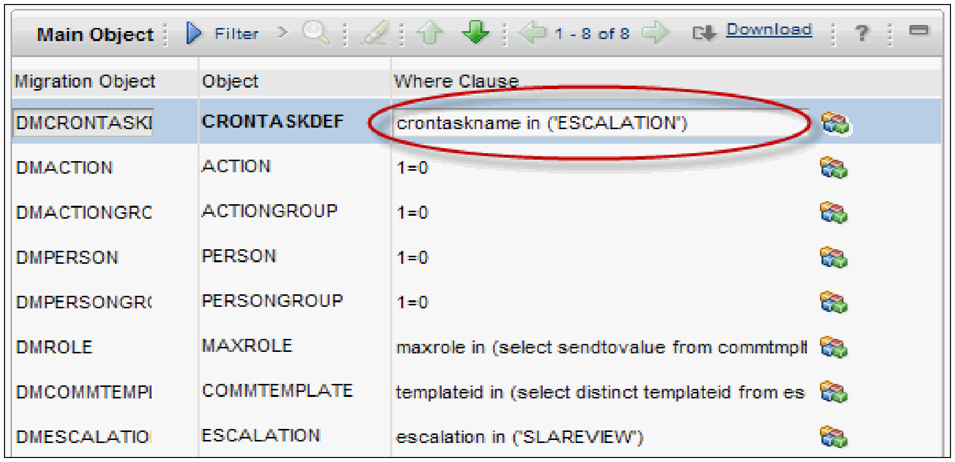
1. In the Source production environment, add the DMCRONTASKDEF object structure to the created migration group that supports escalation migration. Reorder the object structures in the migration group so that DMCRONTASKDEF becomes the first object structure, as shown in Figure 7-24.



Migration Object	Description	Migration Object Order
DMCRONTASKI	Cron task migration object for Migration Manage	1
DMACTION	Action migration object for Migration Manager	2
DMACTIONGRC	Actiongroup migration object for Migration Man	3
DMPERSON	Person migration object for Migration Manager	4
DMPERSONGR(	Person group migration object for Migration Ma	5
DMROLE	Role migration object for Migration Manager	6
DMCOMMTEMPI	Communication template migration object for Mi	7
DMESCALATIO	Escalation migration object for Migration Manag	8

Figure 7-24 Migration group that contains DMCRONTASKDEF object structure

2. Reuse the existing migration package definition, which now reflects the addition of the DMCRONTASKDEF object structure. Figure 7-25 shows the SQL Where clause that is associated with the DMCRONTASKDEF object structure.



Migration Object	Object	Where Clause
DMCRONTASKI	CRONTASKDEF	crontaskname in ('ESCALATION')
DMACTION	ACTION	1=0
DMACTIONGRC	ACTIONGROUP	1=0
DMPERSON	PERSON	1=0
DMPERSONGR(	PERSONGROUP	1=0
DMROLE	MAXROLE	maxrole in (select sendtovalue from commtmplt
DMCOMMTEMPI	COMMTEMPLATE	templateid in (select distinct templateid from es
DMESCALATIO	ESCALATION	escalation in ('SLAREVIEW')

Figure 7-25 SQL criteria to migrate ESCALATION cron task

3. Perform the standard migration steps to create, distribute, and deploy the package.

Upon deployment, all cron task instances that are associated with the ESCALATION cron task are inserted or updated into the Target production database. A significant drawback of this approach is the inability to take only the specific cron task instance that is associated with the escalation that is migrated. By using the Migration Manager, you can specify SQL criteria only against the primary object of an object structure and not on subordinate objects.

Important: If you do not intend to migrate any cron tasks, ensure that you do not collect any cron task configurations by specifying the SQL Where clause 1=0 for the DMCRONTASKDEF object structure.

Best practice: If a business application provides the capability to create or associate configurations automatically, you must use this capability in support of migration. This capability can save time and resources.

Snapshot processing actions and effect on escalations

A Snapshot migration package definition is always associated with a processing action. The processing action determines how the content that is extracted from the Source production environment is processed during deployment into the Target production environment. The default processing action that is associated with a Snapshot migration package definition is Replace. In the Source production environment, the processing action can be changed to AddChange before the package definition is approved or during package creation. Each processing action potentially produces separate results during migration.

Figure 7-26 shows an escalation 1001 that was created in a Source production environment.

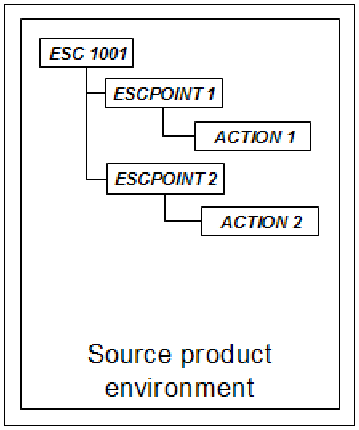


Figure 7-26 Escalation 1001 initially created in a source environment

Escalation 1001 is migrated to the Target production environment. Figure 7-27 shows the result of the migration.

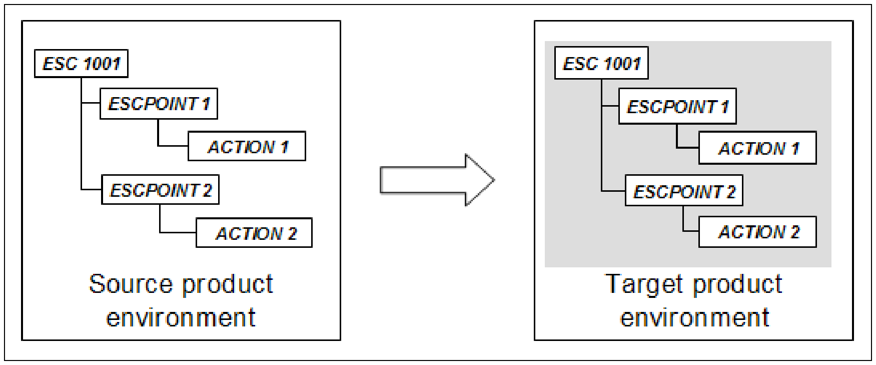


Figure 7-27 Escalation 1001 migrated to the target environment

In the Target production environment, a new escalation point is added to escalation 1001. Figure 7-28 shows the updated escalation.

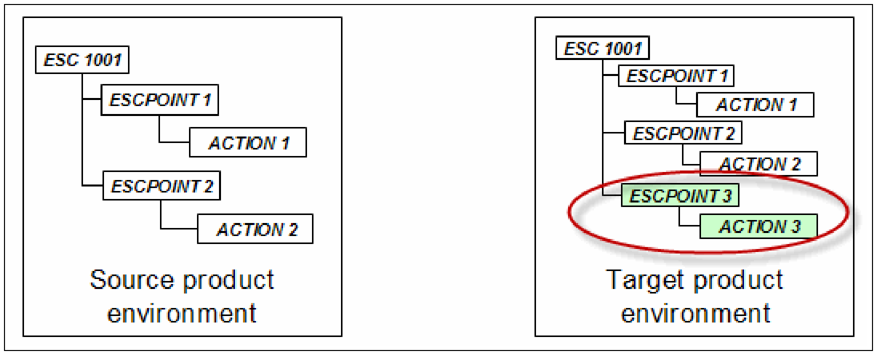


Figure 7-28 Escalation 1001 modified directly in the target environment

In the Source production environment, a new escalation point is added to escalation 1001. Figure 7-29 shows the updated escalation.

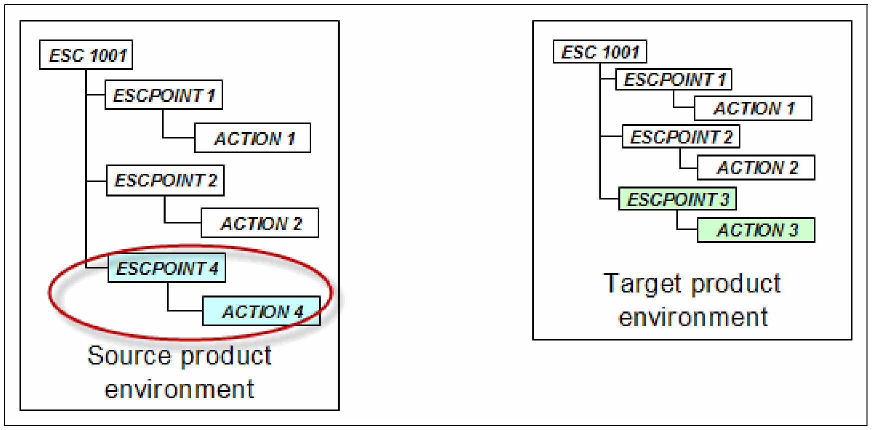


Figure 7-29 Escalation 1001 modified in the source environment

If escalation 1001 is now migrated from Source to Target with a processing action Replace, escalation point ESCPOINT3 in the Target is deleted. The purpose of the Replace processing action is to completely replace the configuration in the Target production environment with the content from the migration package. Figure 7-30 shows the effect of the Replace processing action.

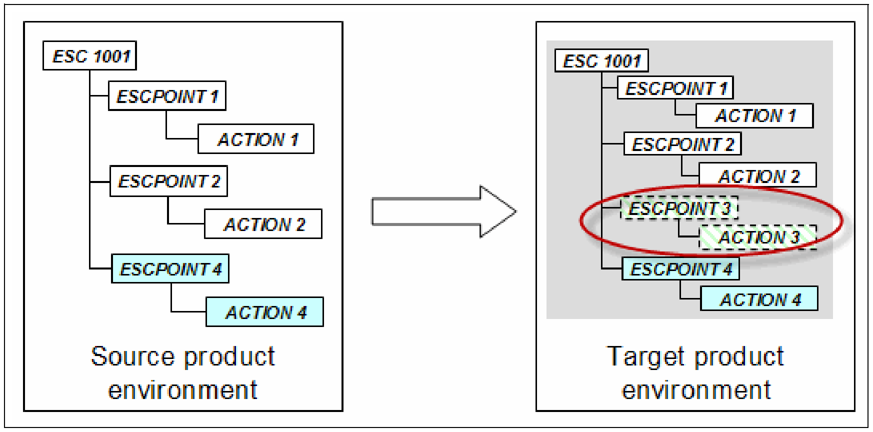


Figure 7-30 Modified escalation 1001 migrated with the Replace processing action

This effect is not the wanted result. Changes that are made directly in a Target production environment, such as a production environment, must be preserved. If escalation 1001 is migrated from the source environment to the target environment with processing action AddChange, escalation point ESCPOINT3 is preserved. The purpose of the AddChange processing action is to leave intact the elements of the configuration in the Target production environment. Figure 7-31 shows the effect of the AddChange processing action.

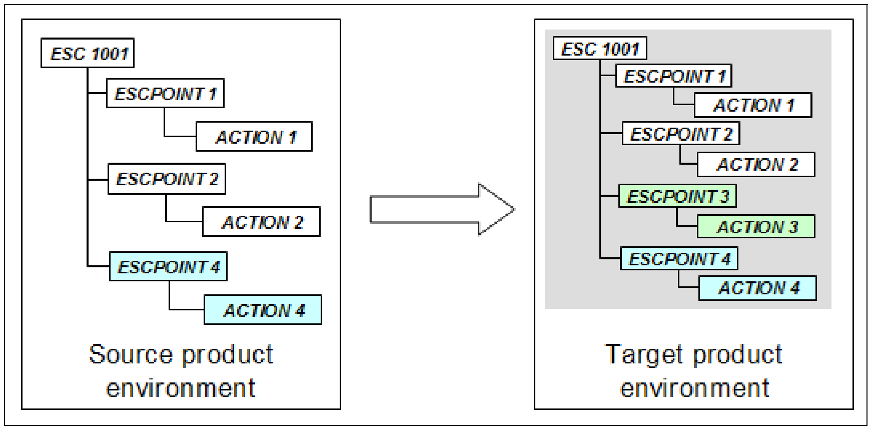


Figure 7-31 Modified escalation 1001 migrated with the AddChange processing action

Best practice: If you deploy into a live production environment, specify the processing action AddChange for Snapshot migration packages.

Collecting attached documents

Certain configurations might be associated with attached documents. For example, a specific communication template might always include an operations manual document as an attachment. When a notification is sent by using this communication template, the document is delivered to the recipients with the mail notification. During migration, the ability to include the attached document with the configuration content is invaluable.

With the release of Maximo Base Services Fix Pack 7.1.1.6, the DMCOMMTEMPLATE object structure includes the DOCLINKS object that maintains the association between a communication template and its attached documents. This association ensures that the attached document files are automatically collected and included in the migration package.

All attached document files are associated with folder and file name information. This information is stored in the DOCINFO table. When the communication template and its attached documents are migrated, the physical file is copied into the folder structure that is specified in the DOCINFO table. Ensure that the folder structure on the file system is the same for Source and target environments. If the folder structures differ, the file is saved but it cannot be viewed as an attached document from the Communication Templates application.

Best practice: During migration, use the ability to automatically collect an attached document for an object. Ensure that the attached document folder structure on the file system is the same in the Source and Target production environments. This approach is more efficient than organizing and collecting compiled sources for a migration package.

Escalations and workflow processes

Escalations can be designed to start workflow processes through actions. When escalations and actions that start workflow processes are migrated, ensure that the workflow process is migrated before the escalation and action. Use two migration packages. The first package migrates the wanted workflow process; the second package migrates the escalation and action.

Escalations and Java classes

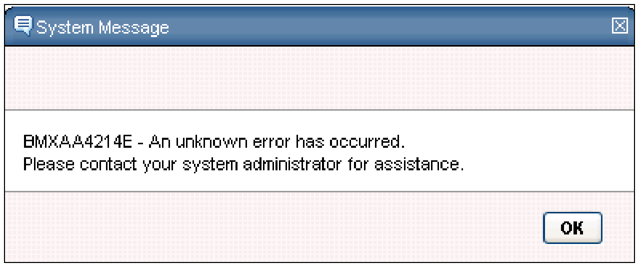
Actions and roles that are associated with an escalation can run custom Java classes. Typically, Java classes are managed in a designated folder structure on a file system in a product development or build environment. A source control system is used to manage multiple versions of the Java class and to control access to the Java source code.

Java classes can be collected in a migration package in the form of compiled sources. When the package is deployed, the Migration Manager prompts the user to download the Java classes in the migration package to a folder from which the Java deployment Enterprise Archive (EAR) can be built. The Migration Manager does not support any build process or deployment of an EAR file into an application server.

An alternative approach is to directly obtain the wanted Java classes from the source control system and run the build steps.

Pitfalls of migrating an active escalation

The migration of an active escalation without the use of either of the approaches that were previously described result in an error-free deployment. However, the escalation cannot be successfully activated. The Escalations application reports error BMXAA4214E, as shown in Figure 7-32.



BMXAA4214E - An unknown error has occurred.
Please contact your system administrator for assistance.

Figure 7-32 Error when an escalation is activated following the migration

The product log file or system console presents a Java stack trace as shown in the following example:

```
java.lang.NullPointerException
        at
psdi.app.escalation.Escalation.activateCronTaskInstance(Escalation.ja
va:140)
        at
psdi.app.escalation.Escalation.actDeactEscalation(Escalation.java:104
)
        at
psdi.webclient.beans.escalation.EscalationAppBean.ADESC(EscalationApp
Bean.java:48)
```

This error occurs because the escalation configuration includes a reference to the cron task instance, but the cron task is not migrated. When the escalation is activated, the Escalation application attempts to activate this non-existent cron task instance.

Following the best practice in migrating escalations avoids this pitfall.

7.8 Migrating escalations with collections

In this section, we implement another escalation scenario in which we use the Migration Collections functionality to drive the migration.

This methodology adopts the following new approaches:

- Migrating escalations by using the collections is a two-step process. You must run Validation twice.
- Manipulate the Validation rules to uncover related configurations.

7.9 Requirements

The requirement calls for an escalation that examines service requests (SR) that include a status of RESOLVED. If they remain open in the system for 10 days, the associated action must change the status of the Service Request to CLOSED.

This requirement also calls for a customized action that is run by the escalation. When the escalation runs, the custom action examines the target Service Request record. The action implements business logic by using the Jython language. The logic of the script is as follows (match steps 1 and 2 in the following logic with the callouts that are shown in Figure 7-33 on page 191):

```
1. If the Service Request has NEW status, and its reported priority
is 1, and an asset is specified, then:
2. The script must take the system generated contact date and the
system generated target start date and increment it by one day each;
The script must create a new work log in the Work Log tab of the
Service Request
```

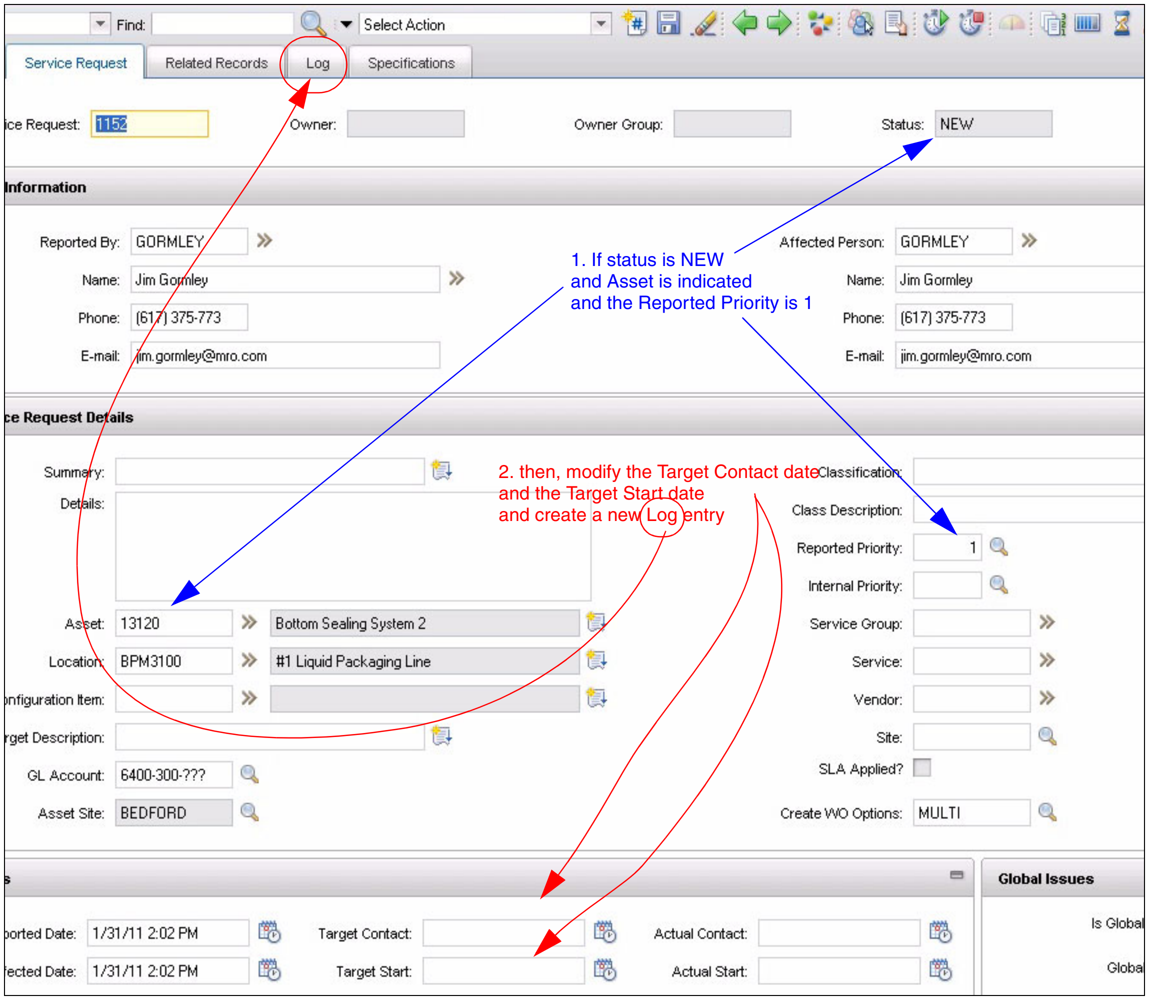
Important: For more information about the Tivoli's process automation engine's scripting capabilities and the Jython programming language, see the following resources:

- Tivoli's process automation engine scripting cookbook:

  http://ibm.co/pPl32E

- Jython programming language:

  http://www.jython.org



ice Request: 1152
Owner:
Owner Group:
Status: NEW

Reported By: GORMLEY
Name: Jim Gormley
Phone: (617) 375-773
E-mail: jim.gormley@mro.com

Affected Person: GORMLEY
Name: Jim Gormley
Phone: (617) 375-773
E-mail: jim.gormley@mro.com

1. If status is NEW
and Asset is indicated
and the Reported Priority is 1

Summary:
Details:
Asset: 13120 Bottom Sealing System 2
Location: BPM3100 #1 Liquid Packaging Line
nfiguration Item:
rget Description:
GL Account: 6400-300-???
Asset Site: BEDFORD

2. then, modify the Target Contact date
and the Target Start date
and create a new Log entry

Classification:
Class Description:
Reported Priority: 1
Internal Priority:
Service Group:
Service:
Vendor:
Site:
SLA Applied?
Create WO Options: MULTI

ported Date: 1/31/11 2:02 PM
fected Date: 1/31/11 2:02 PM
Target Contact:
Target Start:
Actual Contact:
Actual Start:

Figure 7-33 Customized action is run against the service request record

7.10 Solution

The solution is to create and populate a migration collection. Collection validation uncovers related data, such as the escalation action. The validation is run twice: the first validation uncovers the Action Group, the second validation uses the discovered Action Group to uncover the Action that you created. This result is seen because escalations are always associated with an action group and not individual action. In addition to uncovering actions, we implement a Related Data rule to find the Action launchpoint that is associated with the automation script.

7.11 Configuration applications

The following sections describe how to create the escalation.

7.11.1 Creating an action launchpoint

Generally, all work flows or escalations require an action to run to complete that workflow or escalation. You can use SQL expressions to manipulate your actions. However, in some cases you might require scripting applications.

When you create an action launchpoint, the system creates an action record in the Action application.

Complete the following steps to create an action launch point:

1. Open the Automation Scripting application be clicking **System Configuration** → **Platform Configuration** → **Automation Scripts**.
2. From the Select Action menu, click **Create** → **Script with Action Launch Point**.

3. Specify the following values:
   a. Script: `SRDDACTION`
   b. Description: `Automated way of populating target and finish dates in SR` (copy paste this description in all subsequent description fields).
   c. Script language: `Jython`
   d. Click **New Row** in the Variables table window and enter the following variables that are listed in Table 7-1 (all variable names are in lowercase, just as they appear in the script). For the OUT type variables, also select the **Suppress Access Control** option. Click **Next**.

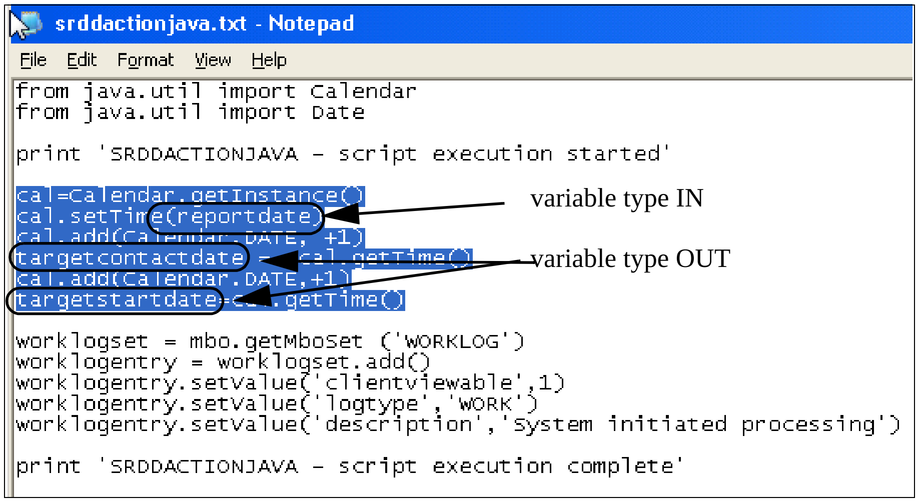
Table 7-1 Variable types that are used in the action script

Variable	Var Type	Binding Type	Launchpoint Attribute
reportdate	IN	ATTRIBUTE	REPORTDATE
targetcontactdate	OUT	ATTRIBUTE	TARGETCONTACTDATE
targetstartdate	OUT	ATTRIBUTE	TARGETSTART

4. Enter the following script or obtain it from this book's FTP site. For more information about how to download the script, see Appendix A, "Additional material" on page 445:

```
from java.util import Calendar
from java.util import Date

print 'SRDDACTIONJAVA - script execution started'

cal=Calendar.getInstance()
cal.setTime(reportdate)
cal.add(Calendar.DATE, +1)
targetcontactdate =  cal.getTime()
cal.add(Calendar.DATE,+1)
targetstartdate=cal.getTime()

worklogset = mbo.getMboSet ('WORKLOG')
worklogentry = worklogset.add()
worklogentry.setValue('clientviewable',1)
worklogentry.setValue('logtype','WORK')
worklogentry.setValue('description','System initiated processing')

print 'SRDDACTIONJAVA - script execution complete'
```

Figure 7-34 shows the variables that are used in the script.



```
from java.util import Calendar
from java.util import Date

print 'SRDDACTIONJAVA - script execution started'

cal=Calendar.getInstance()
cal.setTime(reportdate)
cal.add(Calendar.DATE, +1)
targetcontactdate = cal.getTime()
cal.add(Calendar.DATE,+1)
targetstartdate=cal.getTime()

worklogset = mbo.getMboSet ('WORKLOG')
worklogentry = worklogset.add()
worklogentry.setValue('clientviewable',1)
worklogentry.setValue('logtype','WORK')
worklogentry.setValue('description','System initiated processing')

print 'SRDDACTIONJAVA - script execution complete'
```

Figure 7-34 Variables that are used in the script

5. Click **Create**.

7.11.2 Creating an escalation

Complete the following steps to create an escalation:

1. Open the Escalation application by clicking **System Configuration** → **Platform Configuration** → **Escalations**.
2. Create a record and specify the following values:
   a. Escalation: `SRDDESC`
   b. Applies to: `SR`
   c. Description: `Escalation using Action LP`
   d. Condition:

      ```
      assetnum is not null and reportedpriority = 1 and status = 'NEW'
      and targetcontactdate is null and targetstart is null
      ```

   e. Schedule: `15` seconds
3. Click **New Row** in the Escalations Point and leave it blank. We are not using time measurements from an escalation point.

4. In the Actions tab, click **New Row**.

   The Sequence 10 in the Actions table is the first action. You can run multiple actions and you can choose which action to run first. If you add an action, the sequence is 20.

   The action group number was generated automatically. The business logic assumes that you might want to group your actions in separate groupings and run them per chosen group.

   Escalations typically use notifications, communications templates, and roles for a complete escalation task to fulfill its purpose. In this simplified example, you do not need to configure any other features.

5. Select the SRDDACTION action that was created as a launch point.
6. Save and activate the escalation.

7.11.3 Testing the escalation

Complete the following steps to test the escalation:

1. Open a new service request record (the power application, not self service). The status is NEW and one of the conditions is already met.
2. Specify Maxadmin in the **Reported By** field.
3. For Reported priority value, specify: `1`
4. Click **Submit**.
5. Select an asset from the demonstration database, for example, ITAM2093.
6. Save the record.
7. Refetch the record from the List tab to refresh it. The Target Contact date is 24 hours after the Reported Date and the Target Start date is 48 hours after the Report Date. There also is an entry in the Log tab.

7.11.4 Preparing the escalation for the production environment

Complete the following steps to prepare the escalation for the production environment:

1. Retrieve the escalation that you created earlier (SRDDESC).
2. Deactivate the escalation.

   It is always best practice to deactivate work flows and escalations before they are migrated to the target environment.
3. Duplicate the escalation and rename it by using your first name.
4. Save the record as is.

   Deploy the escalations while it is inactive; otherwise, you must include the corresponding cron task within the package as Order Number 1. The migration group for the cron task is in the SYSTEM (DMCRONTASKDEF) migration group.

7.12 Creating a migration collection

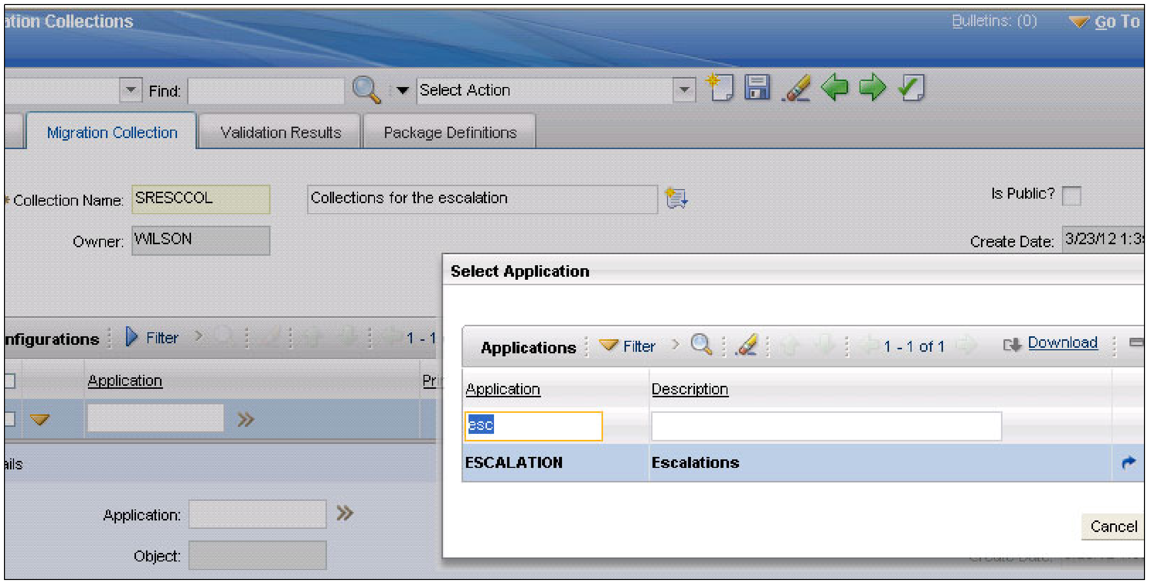
You are now ready to create the migration collections.

1. Open the Migrations Collections application by clicking **System configuration** → **Migration** → **Migration Collections**.

   Enter the following information:

   – Collection Name: SRESCCOL
   – Description: Collections for the escalation

2. Click **New Row** in the Configurations table window.
3. From the Applications lookup, filter by using `esc` and find the ESCALATION application, as shown in Figure 7-35.



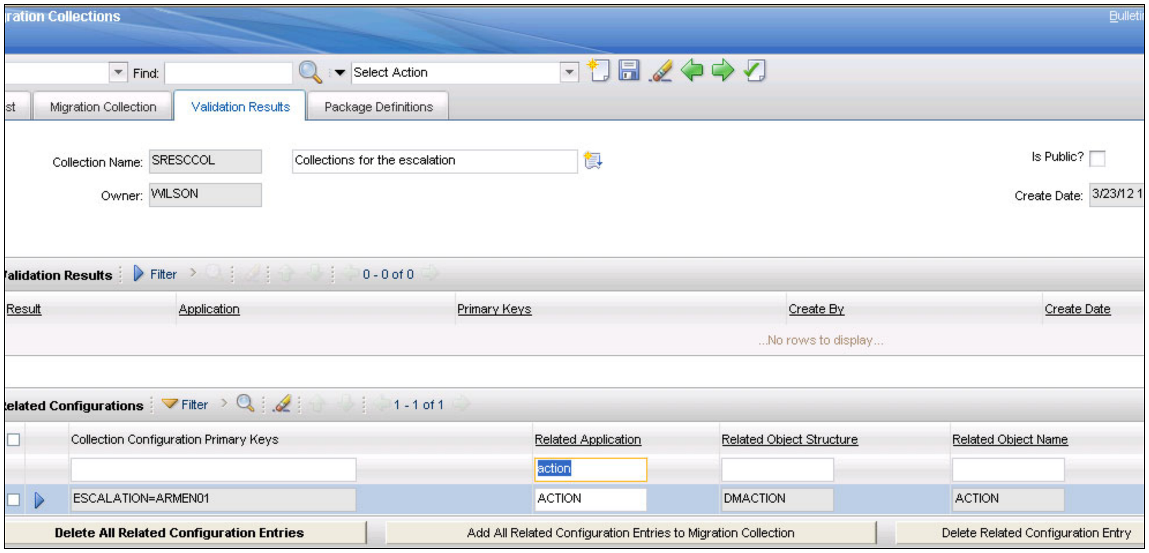
Collection Name: SRESCCOL
Collections for the escalation
Is Public?
Owner: WILSON
Create Date: 3/23/12 1:3
Select Application
Application
Application:
Object:

Application	Description
esc	
ESCALATION	Escalations

Figure 7-35 ESCALATION application

4. Click **Go To** to start the ESCALATION application.
5. Retrieve your escalation (with the eponymous name) and click **Return with Value**.
6. Save the record (you must save it before you validate it).
7. From the Select Action menu, click **Validate Migration Collection**.
8. Click **Start** and then click **Refresh Status** until you see a message that indicates validation is complete.

9. Close the window.
10. Open the filter in the Related Configurations section and press Enter.
11. Page forward. The application of importance here is Action. To ensure that you have all of the related Action applications, filter for `action` in the Related application field.
12. Select the Action configuration and add it to the collection (rightmost button). Save the record, as shown in Figure 7-36.



Collection Name: SRESCCOL — Collections for the escalation

Owner: WILSON

Is Public? ☐

Create Date: 3/23/12 1

Validation Results — 0 - 0 of 0

Result	Application	Primary Keys	Create By	Create Date
...No rows to display...				

Related Configurations — 1 - 1 of 1

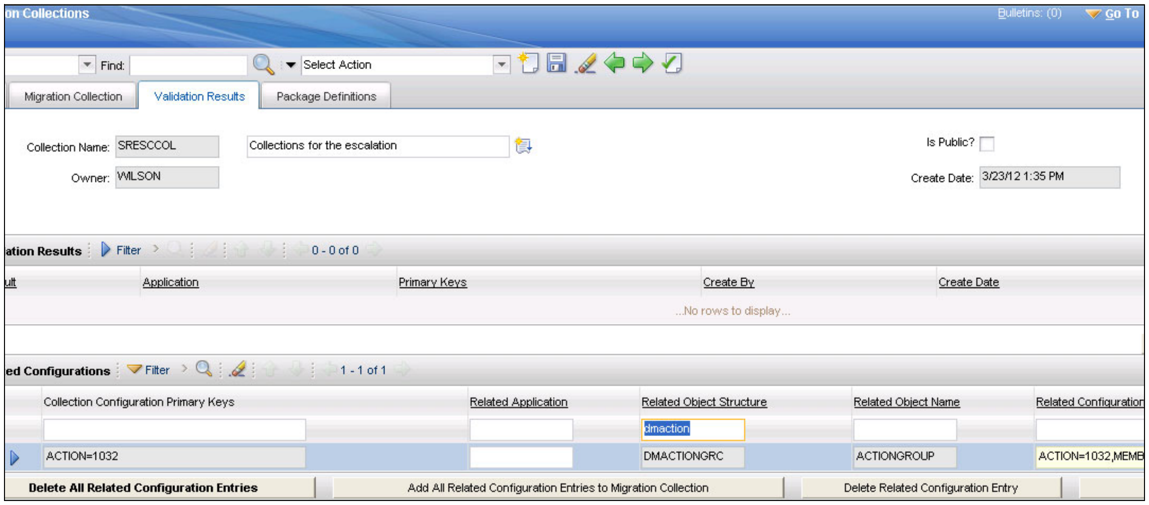
Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name
	action		
ESCALATION=ARMEN01	ACTION	DMACTION	ACTION

ration Collections
Bulleti
Find:
Select Action
st
Migration Collection
Validation Results
Package Definitions
Filter
Delete All Related Configuration Entries
Add All Related Configuration Entries to Migration Collection
Delete Related Configuration Entry

Figure 7-36 Migration Collections

13. After the Action is added to your collection, run validation again. Save the record.

14. Filter by `dmaction` in the Related Object Structure field, as shown in Figure 7-37.



on Collections   Bulletins: (0)   Go To

Find: Select Action

Migration Collection | Validation Results | Package Definitions

Collection Name: SRESCCOL   Collections for the escalation   Is Public?

Owner: WILSON   Create Date: 3/23/12 1:35 PM

ation Results   Filter   0 - 0 of 0

ult	Application	Primary Keys	Create By	Create Date

...No rows to display...

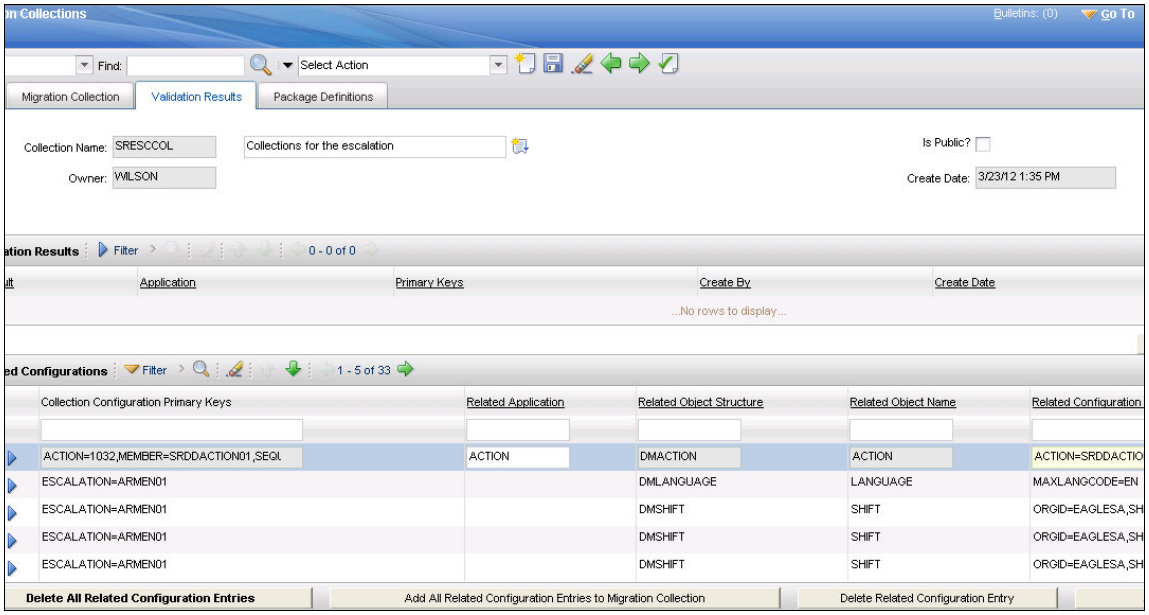
ed Configurations   Filter   1 - 1 of 1

	Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration
			dmaction		
	ACTION=1032		DMACTIONGRC	ACTIONGROUP	ACTION=1032,MEMB

Delete All Related Configuration Entries | Add All Related Configuration Entries to Migration Collection | Delete Related Configuration Entry

Figure 7-37   Filter by dmaction

15. Add the newly discovered related record for the Action application to the collection, as shown in Figure 7-38.



on Collections   Bulletins: (0)   Go To

Find: Select Action

Migration Collection | Validation Results | Package Definitions

Collection Name: SRESCCOL   Collections for the escalation   Is Public?

Owner: WILSON   Create Date: 3/23/12 1:35 PM

ation Results   Filter   0 - 0 of 0

ult	Application	Primary Keys	Create By	Create Date

...No rows to display...

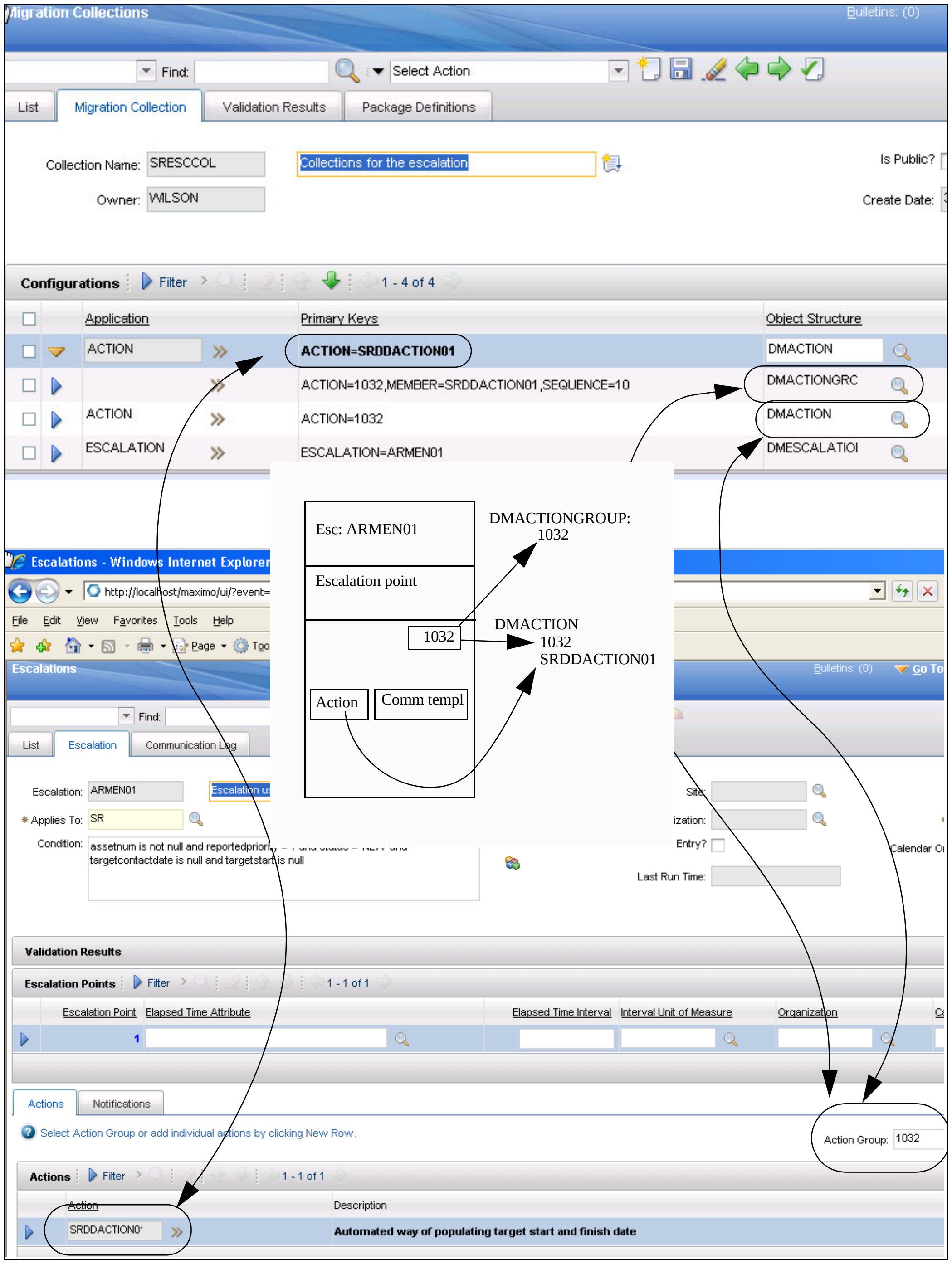
ed Configurations   Filter   1 - 5 of 33

	Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration
	ACTION=1032,MEMBER=SRDDACTION01,SEQL	ACTION	DMACTION	ACTION	ACTION=SRDDACTIO
	ESCALATION=ARMEN01		DMLANGUAGE	LANGUAGE	MAXLANGCODE=EN
	ESCALATION=ARMEN01		DMSHIFT	SHIFT	ORGID=EAGLESA,SH
	ESCALATION=ARMEN01		DMSHIFT	SHIFT	ORGID=EAGLESA,SH
	ESCALATION=ARMEN01		DMSHIFT	SHIFT	ORGID=EAGLESA,SH

Delete All Related Configuration Entries | Add All Related Configuration Entries to Migration Collection | Delete Related Configuration Entry

Figure 7-38   Add the newly discovered related record for the Action application to the collection

Escalation uses an action group. Multiple actions can be part of the group, and each action within the group can be sequenced to occur in a certain order. By default, the Escalation application automatically creates an action group when a user inserts an action into the Action subtab of the escalation main tab. The first validation uncovers the Action Group, the second validation uses the discovered Action Group to uncover the Action that you created, as shown in Figure 7-39 on page 201.


Migration Collections

Bulletins: (0)

Find: Select Action

List | Migration Collection | Validation Results | Package Definitions

Collection Name: SRESCCOL — Collections for the escalation

Owner: WILSON

Is Public?

Create Date:

Configurations — Filter — 1 - 4 of 4

	Application	Primary Keys	Object Structure
	ACTION	ACTION=SRDDACTION01	DMACTION
		ACTION=1032,MEMBER=SRDDACTION01,SEQUENCE=10	DMACTIONGRC
	ACTION	ACTION=1032	DMACTION
	ESCALATION	ESCALATION=ARMEN01	DMESCALATIOI

Escalations - Windows Internet Explorer

http://localhost/maximo/ui/?event=

File Edit View Favorites Tools Help

Escalations — Bulletins: (0) — Go To

Find:

List | Escalation | Communication Log

Escalation: ARMEN01 — Escalation us

Applies To: SR

Condition: assetnum is not null and reportedpriority = 1 and status = NEW and targetcontactdate is null and targetstart is null

Site:

ization:

Entry?

Calendar Or

Last Run Time:

Validation Results

Escalation Points — Filter — 1 - 1 of 1

Escalation Point	Elapsed Time Attribute	Elapsed Time Interval	Interval Unit of Measure	Organization
1				

Actions | Notifications

Select Action Group or add individual actions by clicking New Row.

Action Group: 1032

Actions — Filter — 1 - 1 of 1

Action	Description
SRDDACTION01	Automated way of populating target start and finish date

Figure 7-39 The second validation uncovers the ACTIONGROUP 1032

16. In the Migration Collection tab, click **New Row** and manually add the Autoscript application and the record that is related to the escalation (SRDDACTION).

17. Save the record. (The Save button might be grayed out indicating that the record is saved automatically).

7.13 Using Related rules to find the Action launch point

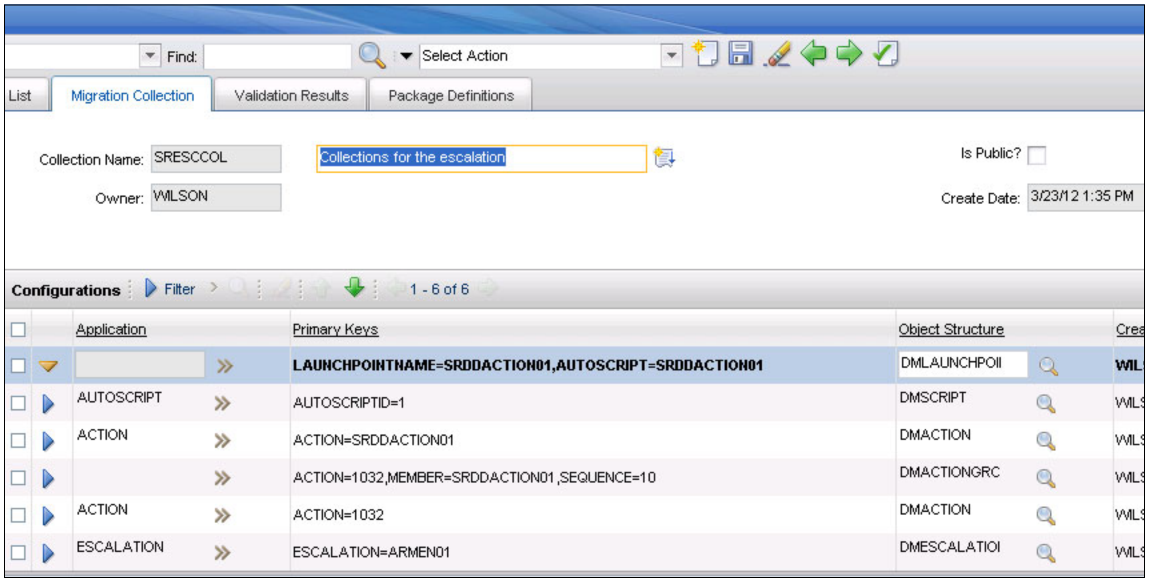
The action that is used in this escalation scenario uses an automation script to implement the business logic. Any automation script is linked with a configuration called a *launch point*. The launch point determines when and how the script is run in the product environment. For this scenario, both the action launch point and the corresponding automation script must be included in the collection to ensure completeness. The automation script can be added directly into the collection by using any of the available collection techniques that are described in Chapter 3, "Migration collections" on page 49. However, these standard techniques cannot be used to collect the launch point. A Related Data rule must be set up to capture the launch point that is associated with the automation script.

Complete the following steps to set up a Related Data rule:

1. From the select Action menu, click **Define Related Data and Lookup Rules**.
2. Click **New Row** and specify the following parameters (as shown in Figure 7-40 on page 203):
   - Rule Name: `LCHPOINT`
   - Primary Object: `AUTOSCRIPT`
   - Related Object: `SCRIPTLAUNCHPONT`
   - Enabled: Selected
3. In the Related Columns field, click **New Row**.

   Enter the following information:

   - For Primary column, specify: AUTOSCRIPT
   - For Related column, specify AUTOSCRIPT
4. Click **OK** and save the record.



Find:

Select Action

List | Migration Collection | Validation Results | Package Definitions

Collection Name: SRESCCOL Collections for the escalation

Owner: WILSON

Is Public?

Create Date: 3/23/12 1:35 PM

Configurations Filter 1 - 6 of 6

Application	Primary Keys	Object Structure	Crea
	LAUNCHPOINTNAME=SRDDACTION01,AUTOSCRIPT=SRDDACTION01	DMLAUNCHPOII	WIL
AUTOSCRIPT	AUTOSCRIPTID=1	DMSCRIPT	WIL
ACTION	ACTION=SRDDACTION01	DMACTION	WIL
	ACTION=1032,MEMBER=SRDDACTION01,SEQUENCE=10	DMACTIONGRC	WIL
ACTION	ACTION=1032	DMACTION	WIL
ESCALATION	ESCALATION=ARMEN01	DMESCALATIOI	WIL

Figure 7-40   Migration Collection record now includes six configurations

After the Related Data rule is saved, you can run validation again. This time, the launch point that is associated with the automation script is returned in the Validation Results tab and can be added to your collection. The final collection includes six configurations, as shown in Figure 7-40.

7.14 Deployment

Complete the following steps to deploy the migration package on the target product environment:

1. Generate the package definition from the Migration Collections application.
2. Browse to the Migration Manager application and approve, create, distribute the package.
3. Access the target product environment to deploy the package.

Admin Mode is not required for this migration scenario. The escalation must be activated from the Escalations application to become functional.

8

Migrating applications and Start Centers

The product installs a rich set of standard application user interfaces (UI) that enables users to perform most of the tasks that are needed to accomplish their jobs. However, sometimes modifications are necessary to simplify the UI experience or to comply to business process rules.

This chapter provides use cases for determining the migration content that is related to application user interfaces (UIs), its related menu items when applicable, and the tasks that are involved in achieving the goal of promoting it to another environment.

Each case covers the most common scenarios. In certain situations, the scope of the modifications involves the use of other tools to migrate, in addition to the Migration Manager.

This chapter includes the following sections:

- Migrating basic application changes by using snapshot
- Migrating basic application changes by using collections
- Migrating complex application changes by using collections
- Migrating a Start Center and a query by using collections

8.1 Migrating basic application changes by using snapshot

When an application is started, a rendered window presentation of the application UI is displayed for user interaction. In this use case scenario, we describe the migration of window presentation modifications in the application UI by using only the Application Designer tool, which is accessed by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**.

8.1.1 Requirements

This use case covers the migration of a new application, which is a clone of the Work Order Tracking application. To analyze the migration requirements, we first review the hypothetical business needs that lead to the change requirements. For example, the application UI needed the following simple modifications:

- The Work Order Tracking application must be cloned.
- The Failure Reporting window is not needed and should be removed.
- The Work Type field must be called Type.
- The Classification field is always required.
- The License field is not used and must be removed from the Work Order window and from the More Search Fields in the Advanced Search menu.
- Add the Report Date in the list window between the Description and Location, and make it sortable.
- Initially sort the searching results list by the Report Date field.
- The signature option for Apply SLA must not be visible to security.
- When the Create KPI toolbar button is clicked when a listing result is displayed, the system must warn that the action affects all listed records.
- Make the Run Report menu option the first item in the Select Action menu.
- Disable the Change Status button in the list window and make it available only in the Work Order window.
- Remove the toolbar buttons for Initiate, Approve, Complete, and Close work order.
- Make the Where clause option first in the Advanced Search menu.

Complete the following steps to change the application UI:

1. Using the Application Designer, search the application Work Order Tracking and open it for modifications. The name of the application is WOTRACK.
2. Duplicate the WOTRACK application and give the new application a name and a description. In this example, we choose WOTRACKR1as a name and Work Order Tracking - IBM Redbooks example1 as a description.

   Important: The commonly used term by the product implementers when duplicating applications is *cloning*. That is, the WOTRACKR1 is a clone of the WOTRACK application.

3. Using the workspace in Application Designer, we made the following changes to the WOTRACKR1 application to meet the requirements:
   a. In the List tab window, add a table column control for the existing object attribute Report Date, between the Description and Location columns. You can change the layout of the control in the window by using editing techniques, such as click and drag, or cut and paste.
   b. Edit the properties of the Report Date control just added, clearing the Sortable? attribute and making the Input Mode attribute Read only.

   Important: The most commonly modified controls are text boxes, check boxes, sections, static text, help grid, table, and push buttons.

   c. Remove the unwanted control for the License field from the Main tab by deleting the control.
   d. Modify the signature option CREATEKPI advanced signature options, selecting the second option Warning appears when this action is selected.
   e. Modify the visibility of the key value APPLYSLA of the Select Action menu option by clearing the Visible? option.
   f. Modify the visibility of the key value STATUS in the toolbar menu options by making the toolbar button only available in the MAIN tab.
   g. Modify the position of the key value SEARCHWHER in the Search menu options by changing the value of sub position to a number lower than the first key value in the list.

Important: This publication does not describe the process that is used to modify each solution because the solution depends on the details of every specific requirement. For more information about how to use the Application Designer tool, see the *IBM Maximo Asset Management 7.5 Developing Applications* at this website:

`http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/pdfs/pdf_mbs_devapps.pdf`

These changes are considered minor in the sense that only the Application Designer tool, which is provided by the product, is used to fulfill the change requirements.

8.1.2 Solution

After the modifications to the WOTRACKR1 application are reviewed, an approach is identified so that all of the related configuration content is migrated by using a single Snapshot migration package.

From a content perspective, the migration of the new application requires the migration of the application's configuration.

8.1.3 Configuration applications that are used for this solution

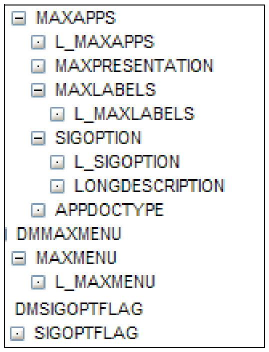
Application configurations are accessed by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**.

8.1.4 Object structures

The following standard object structures that are provided with the product are used to support the migration of the solution that is defined in this use case scenario:

- DMMAXAPPS
- DMMAXMENU
- DMSIGOPTFLAG

Figure 8-1 shows the object structure details for these objects. You can access migration object structures by clicking **Go To** → **System Configuration** → **Migration** → **Object Structures**.



```
MAXAPPS
  L_MAXAPPS
  MAXPRESENTATION
  MAXLABELS
    L_MAXLABELS
  SIGOPTION
    L_SIGOPTION
    LONGDESCRIPTION
  APPDOCTYPE
DMMAXMENU
MAXMENU
  L_MAXMENU
DMSIGOPTFLAG
SIGOPTFLAG
```

Figure 8-1 Object structures for our case scenario

8.1.5 Migration groups

After the change requirements are clearly identified, the next step is to define and create a group of object structures that are the logical containers of the configuration that is needed to migrate.

The object structures for this migration are in two standard migration groups that are called APPLICATION and APPSECURITY. These groups are shipped with the product to support application and security changes.

The configuration content for the advanced signature options that are defined in the Application Designer is not in the standard APPLICATION group. However, it is in the APPSECURITY group.

These migration groups with their dependencies are not useful for scenarios in which subsets of configurations must be selectively migrated and therefore must be left as they are configured. Figure 8-2 shows the object structure list in these migration groups.

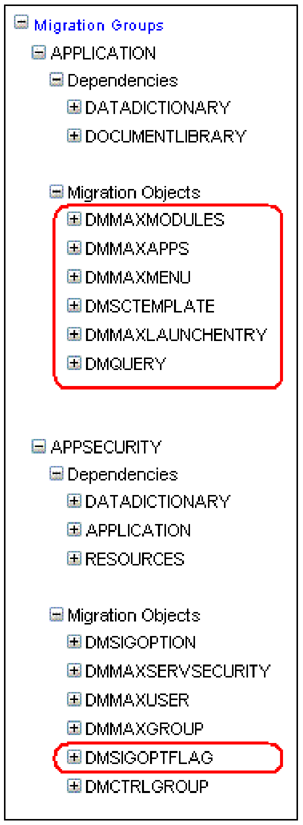
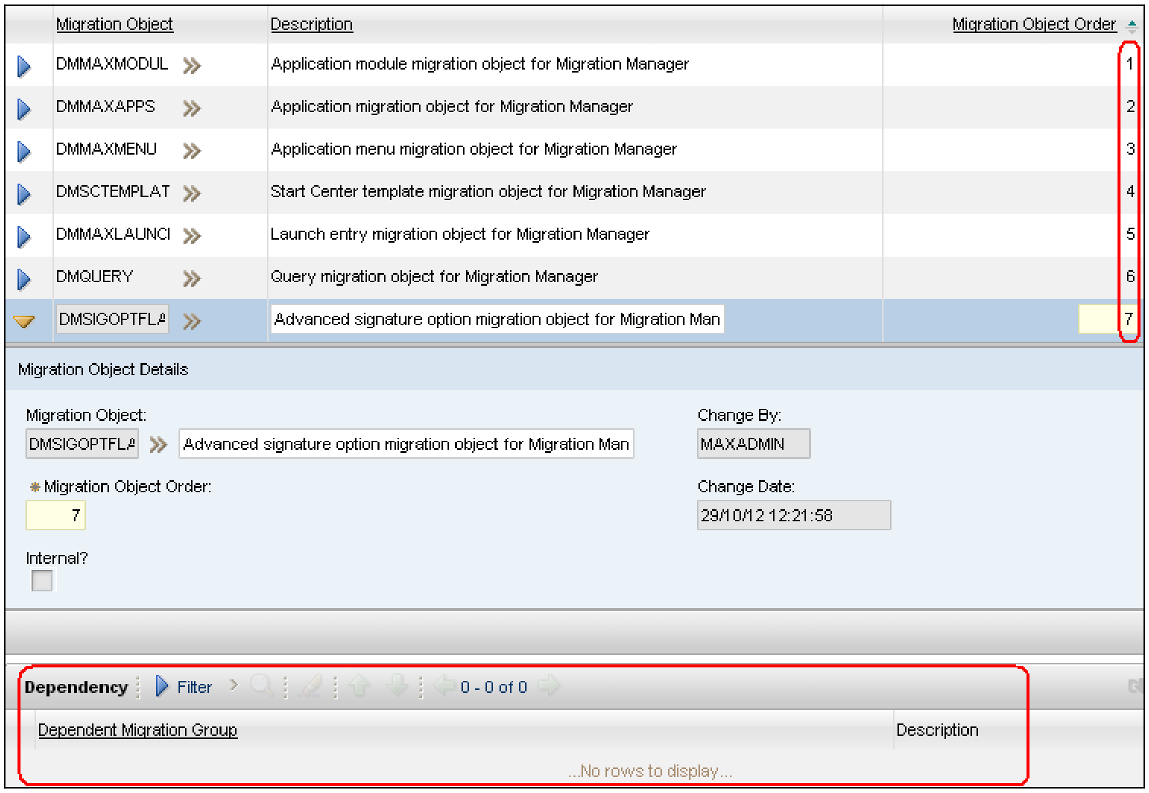

```
⊟ Migration Groups
  ⊟ APPLICATION
    ⊟ Dependencies
      ⊞ DATADICTIONARY
      ⊞ DOCUMENTLIBRARY

    ⊟ Migration Objects
      ⊞ DMMAXMODULES
      ⊞ DMMAXAPPS
      ⊞ DMMAXMENU
      ⊞ DMSCTEMPLATE
      ⊞ DMMAXLAUNCHENTRY
      ⊞ DMQUERY

  ⊟ APPSECURITY
    ⊟ Dependencies
      ⊞ DATADICTIONARY
      ⊞ APPLICATION
      ⊞ RESOURCES

    ⊟ Migration Objects
      ⊞ DMSIGOPTION
      ⊞ DMMAXSERVSECURITY
      ⊞ DMMAXUSER
      ⊞ DMMAXGROUP
      ⊞ DMSIGOPTFLAG
      ⊞ DMCTRLGROUP
```

Figure 8-2 APPLICATION and APPSECURITY migration groups

A new application migration group that includes the object structures that are needed to create a migration package for our use case scenario is necessary. The new group is based on the two groups that are mentioned previously and contains the highlighted object structures as shown in Figure 8-2. By using the Migration Groups application, create a migration group by duplicating the APPLICATION and removing the dependent migration groups.

Now, we must include the object structure that contains advanced signature option configurations in this new group. Add the object DMSIGOPTFLAG to the new group after the DMQUERY object, and then save. Figure 8-3 shows the new group.



Migration Object	Description	Migration Object Order
DMMAXMODUL	Application module migration object for Migration Manager	1
DMMAXAPPS	Application migration object for Migration Manager	2
DMMAXMENU	Application menu migration object for Migration Manager	3
DMSCTEMPLAT	Start Center template migration object for Migration Manager	4
DMMAXLAUNCI	Launch entry migration object for Migration Manager	5
DMQUERY	Query migration object for Migration Manager	6
DMSIGOPTFLA	Advanced signature option migration object for Migration Man	7

Migration Object Details

Migration Object: DMSIGOPTFLA Advanced signature option migration object for Migration Man

Change By: MAXADMIN

Migration Object Order: 7

Change Date: 29/10/12 12:21:58

Internal?

Dependency | Filter | 0 - 0 of 0

Dependent Migration Group	Description

...No rows to display...

Figure 8-3 New migration group MYAPPLICATION

Tip: Migration groups are the shells that are used to define the specific content (or objects) to migrate. They can be reused many times for multiple migration scenarios.

8.1.6 Package definition

After the object structures are identified and organized, the migration process can begin. Create a Snapshot package, which includes the newly defined migration group, to use for defining the specific content to migrate to the target environment.

The next step is to define the SQL criteria within each of the object structures in the group to select the content. The Work Order Tracking cloned application's name, WOTRACKR1, is used to create the criteria in the SQL conditions.

Table 8-1 lists the Where clause SQL command statement conditions for each object. These conditions result in one or many records of configuration content for each migration object.

Tip: A migration package might contain no records for a specific migration object. The Where clause of 1=2 does not generate an error and is commonly used when the object does not contain relevant configuration information in the specific object.

Table 8-1 Set Where clause definitions for the basic application package

MIgration object	Object	Where clause
DMMAXMODULES	MAXMODULES	1=2
DMMAXAPPS	MAXAPPS	app = 'WOTRACKR1'
DMMAXMENU	MAXMENU	moduleapp = 'WOTRACKR1' or keyvalue = 'WOTRACKR1'
DMSCTEMPLATE	SCTEMPLATE	1=2
DMMAXLAUNCHENTRY	MAXLAUNCHENTRY	1=2
DMQUERY	QUERY	1=2
DMSIGOPTFLAG	SIGOPTFLAG	app = 'WOTRACKR1'

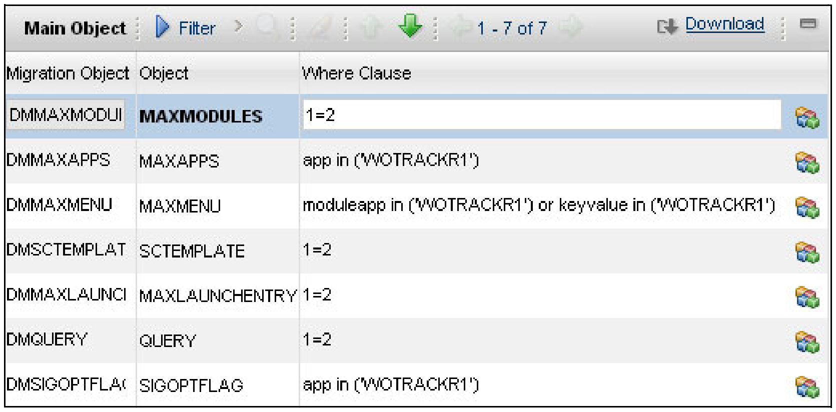
In this case, we migrate only configuration content that is stored in the MAXAPPS, MAXMENU, and SIGOPTFLAG objects. Therefore, the conditions for each object generate the precise content to be stored in the package.

Tip: When more than one application is changed and promoted for migration, you can add more application names that are separated by commas, as shown in the following example:

```
app in (‘WOTRACK’,’CHANGE’,’JOBPLAN’)
```

The Where clauses for DMMAXMODULES, DMSCTEMPLATE, DMMAXLAUNCHENTRY, and DMQUERY are defined with a logical false statement (1=2), which generates no results. That result is the intention in this particular case.

Enter the SQL conditions by clicking the Set Where clause icon for the MYAPPLICATION migration group, as shown in Figure 8-4.


Main Object | Filter | 1 - 7 of 7 | Download

Migration Object	Object	Where Clause
DMMAXMODUI	MAXMODULES	1=2
DMMAXAPPS	MAXAPPS	app in ('WOTRACKR1')
DMMAXMENU	MAXMENU	moduleapp in ('WOTRACKR1') or keyvalue in ('WOTRACKR1')
DMSCTEMPLAT	SCTEMPLATE	1=2
DMMAXLAUNCI	MAXLAUNCHENTRY	1=2
DMQUERY	QUERY	1=2
DMSIGOPTFLA(	SIGOPTFLAG	app in ('WOTRACKR1')

Figure 8-4 Where clause conditions for WOBASICAPP migration package

8.1.7 Deployment

After the package definition is approved and activated, you can create and distribute the package. Then, deploy the package to the target environment. Enable Admin mode before the package is deployed. The Migration Manager requires Admin mode when a package that contains the object MAXAPPS is deployed.

Resource: This publication does not repeat the standard steps that are involved in deploying a package. For more information, see this website:

`http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/index.jsp?topic=%2Fcom.ibm.mbs.doc%2Fgp_migmgr%2Ft_ctr_deploy_packages.html`

8.1.8 Deployment considerations

This use case scenario shows how to migrate simple changes that are made on any application; however, remember the considerations that we describe in the following sections.

Language considerations

For more information about language considerations, see 15.2, “Multiple language considerations” on page 356.

8.2 Migrating basic application changes by using collections

With the introduction of the Migration Collections application in Tivoli's process automation engine release 7.5, the migration of the application UI changes becomes easier and requires minor usage of Structured Query Language in the Migration Collections package definition. In this use case scenario, we use the capabilities of the Migration Collections application to migrate basic window presentation modifications in the application user interface.

8.2.1 Requirements

For this use case scenario, we migrate a clone of the Work Order Tracking - IBM Redbooks example1 application, which was developed in 8.1, "Migrating basic application changes by using snapshot", by using only the Migration Collections. To accomplish this task, we duplicate the WOTRACKR1 application in Application Designer and call it WOTRACKR2.

8.2.2 Solution

We define the steps so that an application definition and all related configurations are migrated in a single collection migration package. From a content perspective, the migration of the cloned application requires the migration of the application, menus, and signature options.

8.2.3 Configuration applications that are used for this solution

Application configurations are accessed by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**.

8.2.4 Object structures

In this use case scenario, we use the following standard object structures that are provided with the product:

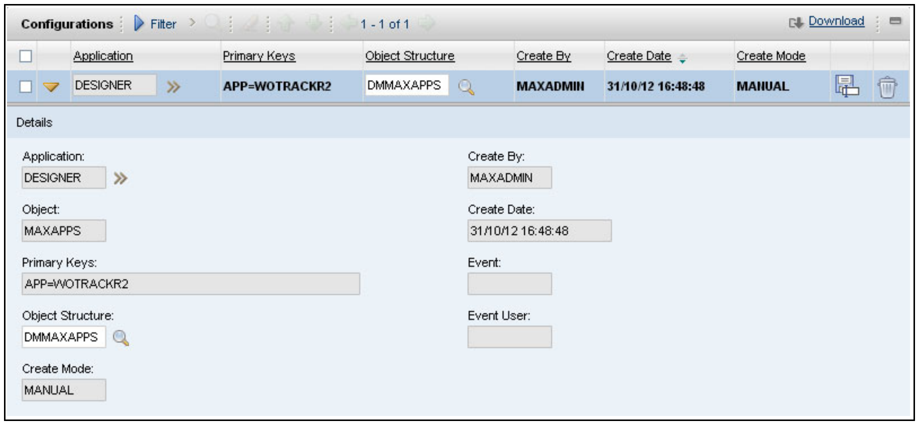
- DMMAXAPPS
- DMMAXMENU
- DMSIGOPTFLAG

8.2.5 Migration Collections definition

After identification and implementation of the requirements in a development environment, we must create a Migration Collection that contains all the data we must migrate the cloned application to a target environment. To prepare the package definition for migration, a collection is defined first and populated with the required system configurations. Then, it is checked for completeness. Afterward, a collection package definition is generated from the collection.

Complete the following steps to create a Migration Collection:

1. Create a migration collection definition and insert the `Work Order Tracking - IBM Redbooks example2` application. By default, the object structure field is set to the ready to use DMMAXAPPS object structure, as shown in Figure 8-5.

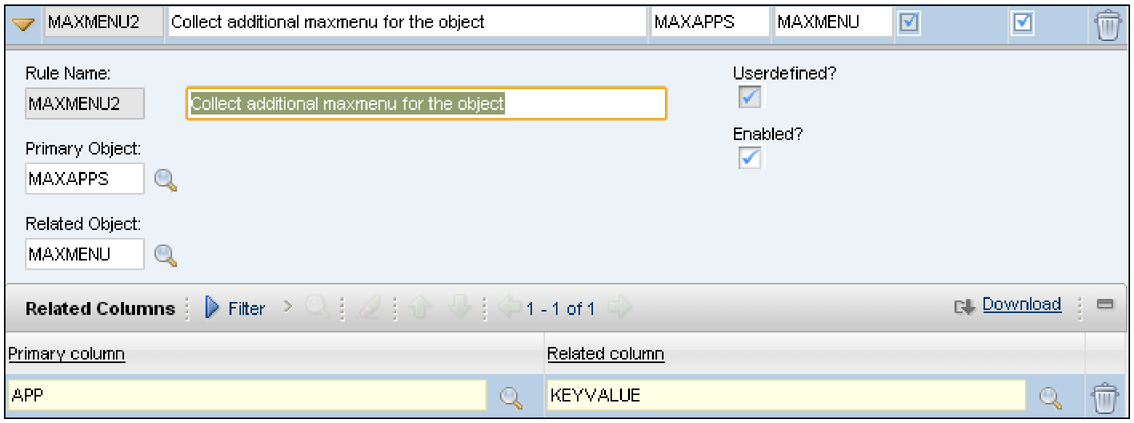


Application	Primary Keys	Object Structure	Create By	Create Date	Create Mode
DESIGNER	APP=WOTRACKR2	DMMAXAPPS	MAXADMIN	31/10/12 16:48:48	MANUAL

Application: DESIGNER	Create By: MAXADMIN
Object: MAXAPPS	Create Date: 31/10/12 16:48:48
Primary Keys: APP=WOTRACKR2	Event:
Object Structure: DMMAXAPPS	Event User:
Create Mode: MANUAL	

Figure 8-5 Migration Collections

2. Add a Related Data rule MAXMENU2 in the Define Related Data and Lookup Rules window. This rule brings all of the MAXMENU records where MAXAPPS.APP = MAXMENU.KEYVALUE. The new rule captures the MAXMENU records that display the clone application in the respective modules (in case the application is attached to more than one module). Figure 8-6 shows the added MAXMENU Related Data rule.



MAXMENU2 | Collect additional maxmenu for the object | MAXAPPS | MAXMENU | ✓ | ✓

Rule Name: MAXMENU2

Collect additional maxmenu for the object

Primary Object: MAXAPPS

Related Object: MAXMENU

Userdefined? ✓

Enabled? ✓

Related Columns | Filter | 1 - 1 of 1 | Download

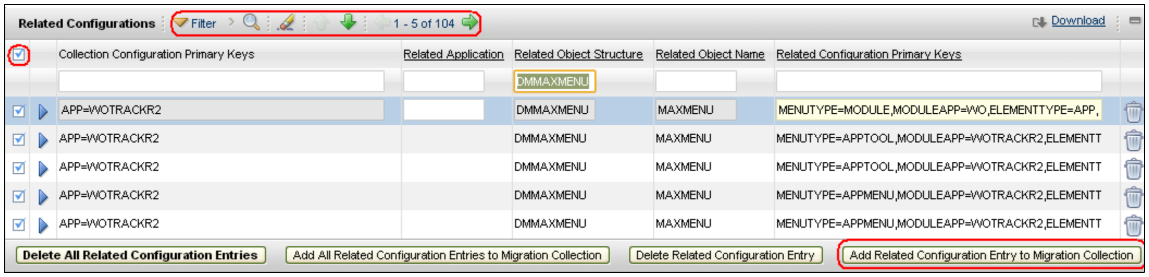
Primary column	Related column
APP	KEYVALUE

Figure 8-6 Another MAXMENU Related Data rule

3. Validate the migration collection definition to find all the related configurations data that we must include in it. The result of the validation contains related configuration records from the following object structures:

   – DMLANGUAGE
   – DMCURRENCY
   – DMDOCTYPES
   – DMLANGUAGE
   – DMMAXDOMAIN
   – DMMAXMENU
   – DMMAXMODULES
   – DMMAXOBJECTCFG
   – DMMAXRELATIONSHIP
   – DMMAXSEQUENCE
   – DMMAXSERVICE
   – DMORGANIZATION
   – DMSETS
   – DMSIGOPTFLAG

In our use case scenario, we must add only DMMAXMENU and DMSIGOPTFLAG object structure-related configuration to the migration collection definition. Figure 8-7 shows a filtered validation result that we must add to the migration collection.



✓		Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
				DMMAXMENU		
✓	▶	APP=WOTRACKR2		DMMAXMENU	MAXMENU	MENUTYPE=MODULE,MODULEAPP=WO,ELEMENTTYPE=APP,
✓	▶	APP=WOTRACKR2		DMMAXMENU	MAXMENU	MENUTYPE=APPTOOL,MODULEAPP=WOTRACKR2,ELEMENTT
✓	▶	APP=WOTRACKR2		DMMAXMENU	MAXMENU	MENUTYPE=APPTOOL,MODULEAPP=WOTRACKR2,ELEMENTT
✓	▶	APP=WOTRACKR2		DMMAXMENU	MAXMENU	MENUTYPE=APPMENU,MODULEAPP=WOTRACKR2,ELEMENTT
✓	▶	APP=WOTRACKR2		DMMAXMENU	MAXMENU	MENUTYPE=APPMENU,MODULEAPP=WOTRACKR2,ELEMENTT

Delete All Related Configuration Entries
Add All Related Configuration Entries to Migration Collection
Delete Related Configuration Entry
Add Related Configuration Entry to Migration Collection

Figure 8-7 Related configuration entries

4. Because we added new configurations to the migration collection in the previous step, run another validation. The new validation results contain related configuration from the DMSIGOPTION object structure that we must add to the migration collection.

   Important: When we add related configuration to the migration collection, we must run validation and check the results for other related configurations that we must add to the migration collection definition package.

5. Create the package definition from the Migration Collections Select action menu, then open the package in the Migration Manager application by clicking the **Go To Migration Manager** icon in the Package Definitions tab.

8.2.6 Deployment

In the migration manager application, we must approve and activate the package definition that was created in the Migration Collections application. Then, we create and distribute the migration package. Before the package is deployed into the target environment, we must take into account that Migration Manager requires Admin mode when a package that contains the MAXAPPS object is deployed.

8.2.7 Deployment considerations

In the following section, we describe deployment considerations for this migration.

8.2.8 Workflow considerations

If we added workflow support to the application in the development environment, we must add it manually to the application after the migration.

8.3 Migrating complex application changes by using collections

In this new use case scenario, we describe the migration of more complex configuration content that involves the use of other applications before and after the use of the Application Designer tool.

A new menu item is created, new library lookups are built, new automation scripts are developed, and security restrictions are applied to data attributes.

This sequence of events in the configuration cycle causes migration dependencies that are analyzed during the migration requirements and solutions. Then, the migration dependencies are addressed during the migration package definition and deployment considerations.

8.3.1 Requirements

For this use case, we migrate modifications that are made for the Assets application. To analyze the migration requirements, we first review the following hypothetical change requirements in which is a need to record test results for serialized assets, track these tests, and send notifications when a test failed:

- A new table is created with a one-to-many relationship from the assets table to this new Test Results table.
- This new table stores information, such as the test result, date, type of test, description of the test, and a defect code if the test failed, for any asset that is serialized.
- The test results are recorded by using a new application UI that is available only to the testing team.

- The test type, defect code, and results are validated against predefined values.
- The date field defaults to the current date when you create an asset test record, and the application accepts only past or current date.
- The test result cannot be deleted or duplicated.
- Searching and listing options are available in the List tab.
- All information is required except for the defect code. A defect code is only required if the test failed.
- A new tab is added to the Asset application to display the test results that are associated to the asset, and is accessible only to the test team.

To fulfill these needs, we make the following minimum change configurations:

1. By using the Domains application, three ALN domains are created to use for the validation values of the results, type of test, and defect code. The names of these domains are MYTESTRESULT, MYTESTTYPE, and MYDEFECTCODE.
2. By using the Domains application, a Table domain is created for looking up values in the linked asset field of the new table. The name of the domain is MYASSETTB.
3. By using the Database Configuration application, a table object is created with all the attributes required, new domain associations, and a new relationship to the ASSET table, which is used to display the asset serial number and other object definitions. The name of the new table object is MYTESTS.
4. By using the Application Designer application, a table ID reference to the table domain created in prior steps is created in the LOOKUPS system library XML.
5. By using the Application Designer application, an application type that is called Power App is created with all of the configuration data that is required for the new table object. The name of the new application is MYTESTS.
6. By using the Application Designer application, the existing Assets application is configured with a new tab and controls that are needed to display the test results records that are linked to the asset. A signature option is created to configure the security restrictions for this new tab. The name of the existing application is ASSET.
7. By using the Security Groups application, two security groups are created for signature security access configuration to the new Test Results application and the new tab in the Assets application. The names of the two new security groups are TESTMGR and TESTTECH.

8. By using the Conditional Expression Manager application, a condition is created that evaluates the test result attribute when the value is FAILED. The name of the new conditional expression is MYFAILED.
9. By using the Security Groups application, a global data restriction is created to evaluate the test result field during data entry, which is based on the condition that was created in the previous step, and to dynamically change the input mode to Required of the test defect code field. The attribute restriction is for the application name MYTEST.
10. By using the Automation Scripts application, a script MYTESTDATE with attribute launchpoint is created. This script is used to validate the date field and displays a message if the field is set to a future date.
11. By using the Database Configuration application, a message is created. This message is used in the date field validation automation script that was created in the previous step.

Important: The configuration content that is generated from at least six separate applications must be migrated. The order in which the configuration is created is important.

8.3.2 Solution

After the requirements for this case scenario are reviewed, we identified an approach so that all the related configuration content is migrated by using one Migration Collection package.

From a content perspective, the migration of the modified application requires the migration of the following configurations in order:

- Domains
- Database table, sequence, relationships, and messages
- Conditional expressions
- System library lookups
- Applications, menus, and signature options
- Automation scripts
- Global security restrictions
- Security groups

8.3.3 Configuration applications that are used for this solution

The following configuration applications were used for this solution:

- Domains

  To access the Domains application, click **Go To** → **System Configuration** → **Platform Configuration** → **Domains**.

- Database table, sequence, relationships, and messages

  To start the Database Configuration application, click **Go To** → **System Configuration** → **Platform Configuration** → **Database Configuration**.

- Conditional expressions

  To access the Conditional Expression Manager application, click **Go To** → **Administration** → **Conditional Expression Manager**.

- System library lookups

  To start the Application Designer application, click **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**.

  From the Application Designer, select **Select Action** → **Export System XML**.

- Applications, menus, and signature options

  These options can be configured from the Application Designer application, which is accessed by clicking **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**.

- Automation scripts

  To access the Automation Scripts application, click **Go To** → **System Configuration** → **Platform Configuration** → **Automation Scripts**.

- Global security restrictions

  To configure global security restrictions, click **Go To** → **System Configuration** → **Security Groups**, and select **Select Action** → **Global Data Restrictions**.

- Security groups

  To start the Security Groups application, click **Go To** → **System Configuration** → **Security Groups**.

- Object structures

  The Object Structures application can be accessed by clicking **Go To** → **System Configuration** → **Migration** → **Object Structures**.

8.3.4 Object structures

Several standard object structures are identified for the migration of the configuration content for this scenario. However, the configuration content of the global data restrictions and the system library lookups requires special attention.

Tip: Always try to use the standard objects and groups that are shipped with the product because they reflect the logical business object structures and associations that are built by the developers.

Standard object structures

The following standard object structures that are provided with the product are used to support the migration requirements:

- DMMAXOBJECTCFG
- DMMAXSEQUENCE
- DMMAXDOMAIN
- DMMAXRELATIONSHIP
- DMMAXLOOKUPMAP
- DMMAXMESSAGES
- DMCONDITION
- DMMAXAPPS
- DMMAXMENU
- DMSIGOPTION
- DMMAXGROUP
- DMMAXINTOBJECT
- DMSCRIPT
- DMLAUNCHPOINT

Global data restrictions content that is related to the application

This configuration content is in the table SECURITYRESTRICT, which is defined in the standard object structure DMMAXGROUP, and its key relationship is a security group. When the data restriction is global, it is not related to a security group.

We cannot migrate this content by using the standard object structure because our use case configured a global data restriction, which is linked to an application name and not to a security group.

DMMAAXGROUP: For more information about the standard migration object structure DMMAXGROUP, see Chapter 6, “Migrating security configuration data” on page 121.

Therefore, to migrate the global security restriction, you must create a migration object structure. The new object is a duplicate of the DMMAXAPPS object, and the table SECURITYRESTRICT is added to relate its content by application name.

Create a migration object structure by duplicating DMMAXAPPS with the following characteristics:

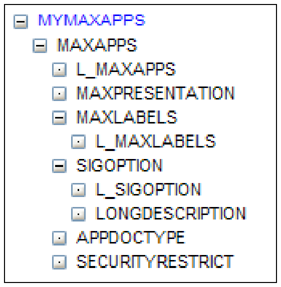
- Object structure: MYMAXAPPS
- Description: Application with data security restriction

Add a source object at the end of the list of source objects for MYMAXAPPS with the following characteristics:

- Object: SECURITYRESTRICT
- Parent object: MAXAPPS
- Relationship: SECURITYRESTRICT
- Object order: 6 (or default)

Save the new object structure.

Figure 8-8 shows the new MYMAXAPPS object structure list.



```
⊟ MYMAXAPPS
  ⊟ MAXAPPS
    ⊡ L_MAXAPPS
    ⊡ MAXPRESENTATION
    ⊟ MAXLABELS
      ⊡ L_MAXLABELS
    ⊟ SIGOPTION
      ⊡ L_SIGOPTION
      ⊡ LONGDESCRIPTION
    ⊡ APPDOCTYPE
    ⊡ SECURITYRESTRICT
```

Figure 8-8 New migration object structure MYMAXAPPS

The new object structure that was created requires the migration of its configuration content, and it is in the standard object structure DMMAXINTOBJECT.

System library lookups object structure

The configuration of system lookups is done by using the Application Designer, and it can be migrated by using the same method in the target environment. However, in this scenario, we create a migration object structure to include in our migration group.

This configuration is stored in the MAXPRESENTATION table in the form of XML content. This table also contains the XML representations of every application and the table is in the MYMAXAPPS object structure, as shown in Figure 8-8 on page 223. Because there is no application that is built for the LOOKUPS library, its content is identified by the application name in the presentation table.

Create a migration object structure that is called MYMAXPRESENTATION with all of the default values and the following characteristics:

- Object structure: MYMAXPRESENTATION
- Description: XML presentations
- Consumed by: MIGRATIONMGR

Add a source object at the end of the list of source objects for MYMAXPRESENTATION with the following characteristics, leaving all of the default values:

- Object: MAXPRESENTATION
- Object order: 1 (or default)

Save the new object structure.

Figure 8-9 shows the new MYMAXPRESENTATION object structure.

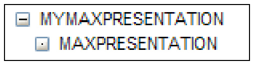

MYMAXPRESENTATION
MAXPRESENTATION

Figure 8-9 MYMAXPRESENTATION object structure

The new object structure that was created also requires the migration of its configuration content, which is migrated by using the standard object structure DMMAXINTOBJECT.

Migration object structures

The following revised list shows the object structures that are used to support the migration requirements, in no particular order:

- DMMAXOBJECTCFG
- DMMAXSEQUENCE
- DMMAXDOMAIN
- DMMAXRELATIONSHIP
- DMMAXLOOKUPMAP
- DMMAXMESSAGES
- DMCONDITION
- DMMAXAPPS
- DMMAXMENU
- DMSIGOPTION
- DMMAXGROUP
- DMMAXINTOBJECT
- DMSCRIPT
- DMLAUNCHPOINT
- MYMAXAPPS
- MYMAXPRESENTATION

Important: For more information about migration object structures, see the *Object Structures documentation* at this website:

http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/index.jsp?topic=%2F
com.ibm.mbs.doc%2Fintobject%2Ft_ctr_work_obj_structs.html.

8.3.5 Migration Collections definition

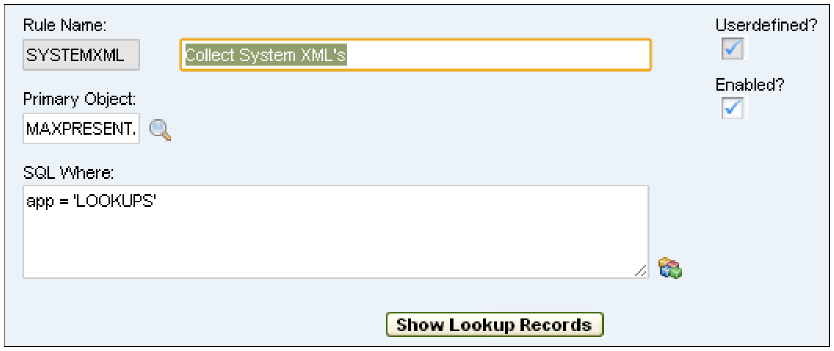
After the change requirements are reviewed and the object structures are identified, the next step is to define the migration collection contents that are used in the migration manager package.

Complete the following steps to create the collection definition:

1. In the Migration Groups application, duplicate the APPLICATION migration group and replace the DMMAXAPPS object structure with the customized MYMAXAPPS object structure.
2. Create a migration group for the MYMAXPRESENTATION object structure.
3. In the Migration Collections application, create a migration collection definition and insert the following configurations:
   - From Database Configuration application, include the MYTESTS and ASSET objects.
   - From Application Designer, include the MYTESTS and ASSET applications. For the MYTESTS application, change the default Object Structure to MYMAXAPPS to migrate the global data restriction.
   - From Conditional Expression Manager applications, include the MYFAILED condition.
   - From the Security Groups application, include the TESTMGR and the TESTECH security groups.
   - From the Automation Scripts application, include the MYTESTDATE script.
   - From the Object Structure application, include the MYMAXAPPS and MYMAXPRESENTATION object structures.

4. In the Define Related Data and Lookup Rules window, we must define the following rules:
   - To migrate the MAXLOOKUPMAP records for the MYTESTS object, we must modify the MAXLOOKUPMAP rule and set the following fields:
     - Rule Name: MAXLOOKUPMAP
     - Description: Collect all the lookupmap for the object
     - Primary Object: MAXOBJECTCFG
     - Related Object: MAXLOOKUPMAP
     - Primary Column: OBJECTNAME
     - Related Column: Target
   - To migrate the Automation Scripts launchpoints, we must add a new Related Data rule and enable it with the following parameters:
     - Rule Name: AUTOSCRIPTS
     - Description: Collect all the launchpoints for the autoscript
     - Primary Object: AUTOSCRIPT
     - Related Object: SCRIPTLAUNCHPOINT
     - Primary Column: AUTOSCRIPT
     - Related Column: AUTOSCRIPT
   - To migrate the MAXMENU records that connect the MYTESTS application to its respective application module, we must enable the rule MAXMENU2 that we added in 8.2.5, “Migration Collections definition” on page 215.
   - To migrate the message that we added in the database configuration application, we must set the where clause in the MAXMESSAGES lookup rule and enable it.
   - To migrate the LOOKUPS System XML, we must add a lookup rule for the MAXPRESENTATION database object. First, we must create a migration group and add to it the MYMAXPRESENTATION object structure. Then, we must add the MAXPRESENTATION object to the DMCOLLLOOKUP domain to use the MAXPRESENTATION object as a primary object in the lookup rule. Finally, we must add a lookup rule and enable it. Figure 8-10 illustrates the added rule.

     Enter the following information to add a lookup rule, as shown in Figure 8-10:
     - Rule Name: SYSTEMXML
     - Description: Collect System XMLs
     - Primary Object: MAXPRESENTATION
     - SQL Where: app = 'LOOKUPS'


Rule Name: SYSTEMXML Collect System XML's

Userdefined? ✓

Enabled? ✓

Primary Object: MAXPRESENT.

SQL Where: app = 'LOOKUPS'

Figure 8-10 System XML's lookup rule

5. Validate the migration collection definition. The validation automatically includes the lookup rules data in the collection definition and we can choose which validation results we must include in the collection definition package. In our use case scenario, we must add the filtered validation results records that are shown in Table 8-2 as a related configuration to the migration collection.

Table 8-2 Related configuration entries to the migration collection

Collection Configuration Primary Keys Filter	Related Object Structure Filter	Related Configuration Primary Keys Filter
OBJECTNAME=MYTESTS	DMMAXDOMAIN	
OBJECTNAME=MYTESTS	DMMAXSEQUENCE	
OBJECTNAME=ASSET	MAXRELATIONSHIP	PARENT=ASSET,NAME=MYTESTS
OBJECTNAME=MYTESTS	DMMAXRELATIONSHIP	
OBJECTNAME=MYTESTS	DMMAXLOOKUPMAP	
APP=MYTESTS	DMMAXMENU	
APP=ASSET	DMMAXMENU	KEYVALUE=MYTESTS
GROUPNAME=TESTMGR	DMSIGOPTION	
GROUPNAME=TESTTECH	DMSIGOPTION	
AUTOSCRIPT=MYTESTDATE	DMLAUNCHPOINT	

Note: The validation fails if the system property `mxe.db.fetchResultStopLimit` is too low. For the migration, it is better to temporarily increase the system property value.

6. After we add the related configuration entries to the migration collection definition in the previous step, run another validation to check whether there are new related configuration entries that we must add to the collection. After the result is checked, we can proceed with the creation of the migration package.
7. In the Define Package Definition Exception window, check that the exception TABLE CROSSOVER DOMAIN is enabled. This exception automatically splits the migration package by two and we can migrate the MYASSETTB domain.
8. Create the migration package definition. Because the collection definition package contains the configuration for the MYASSETTB table domain, the Migration Collection application follows the exception rule in step 5 and creates two migration packages with a package order number.

8.3.6 Deployment

After the two package definitions are created, approved, and activated, create and distribute each package to the target. After the packages are available from the target environment, it is time to deploy them.

For the purposes of this book, we assume that all required changes and release protocols (if any) were put in place. We also assume that the users in the target environment were notified that the system is unavailable during deployment time.

Complete the following steps to deploy the packages:

1. Enable Admin mode from the Migration Manager application.

   Because the configuration content modifies the data dictionary and touches the application presentations for the UI (the MAXAPPS object is in the package), the system requires this mode of operation during deployment. We also recommend that you use the minimum amount of application infrastructure traffic by not allowing any users to connect to the environment.
2. After verifying that the system is now in Admin mode, load and deploy the package with the order number two. This package contains the MYASSETTB table domain and must be deployed first. If an error occurs, stop and follow troubleshooting procedures until the error is corrected. Continue to the next step only if there are no errors.

3. After the first package is deployed without any errors, verify that the content was migrated. Do not proceed until the content is verified.
4. Load and deploy the package with the order number one. This package contains all of the remaining configuration. Again, if errors occur, stop and follow troubleshooting procedures. Continue to the next step only if there are no errors.
5. After all of the content is validated, disable Admin mode in the system so that users can log in.

For more information about troubleshooting, see Chapter 16, “Troubleshooting” on page 419.

8.3.7 Deployment considerations: Admin mode

When packages that contain structural changes are deployed, such as in the data dictionary or application presentation, the system expects the Admin mode operation to be enabled. In a single Java virtual machine (JVM) server environment, there is no problem. However, when you are working on a multiple server environment or a cluster, all of the servers in the cluster must switch to this mode.

During this mode, the application does not allow any users to log in without the administrator’s permission. Always remember to disable this mode after the deployment is complete.

8.4 Migrating a Start Center and a query by using collections

Start Centers are the landing pages that are presented to the user when they log in to the product. There are several Start Centers to which a user can access, based on the security configuration. The layout and content of the Start Center is stored in XML presentation form, in a similar manner to the application UI.

In this case scenario, we review the migration of a new Start Center template, and the application query that is configured for a result set.

8.4.1 Requirements

Previously, we reviewed a case scenario where a new application for test results was created. As part of the same requirement, the following new Start Center is requested to be migrated:

- A new test portal Start Center is created for test managers and technicians and is linked to the identified security groups for these roles.
- The Start Center contains a favorite application portlet with links to the new application Test Results and to the Assets application.
- The Start Center contains the usual bulletin board and task portlets.
- The Start Center contains a result set portlet that contains a new query that was created in the new application Test Results. It also filters the test table to record with the result of Failed.

The following minimum change configurations are made to fulfill the requirement:

1. By using the Test Results application that was created in 8.3, “Migrating complex application changes by using collections” on page 218), create a query to filter Failed results. The clause name for the query is identified as FAILEDTEST.
2. By using the Start Center application, create a template with a Narrow-Wide layout with the Favorite Applications portlet on the left and the Bulletin Board, Inbox Assignment, and Result Set portlets on the right. The Result Set portlet must point to the new query in the Test Results application. The description of the new template is My Failed Asset Test.

Important: In this case scenario, we created the configuration by using the query capabilities of existing applications and the Start Center template configuration. The sequence in which the configuration was created is important to consider during migration.

8.4.2 Solution

After the changes in our scenario are reviewed, our migration approach identifies the use of the migration collection capabilities as the easiest and the fastest way to migrate all the start center configuration content.

From a content perspective, the migration of the new start center template requires the migration of the QUERY and the Start Center Template.

8.4.3 Configuration applications that are used for the solution

We show the Configuration applications that were used for this solution and how you can access these applications.

Queries for the Test Results application

To create queries, click **Go To** → **Assets** → **Test Results**, and then click **Save Query**.

Start Center templates

You can modify or create new Start Center templates by clicking **Start Center** → **Create New/Modify Template**.

8.4.4 Object structures

With the introduction of the Tivoli's process automation engine 7.5, a new standard object structure for the QUERY object is supplied with the product. For our use case scenario, we use the following standard object structures:

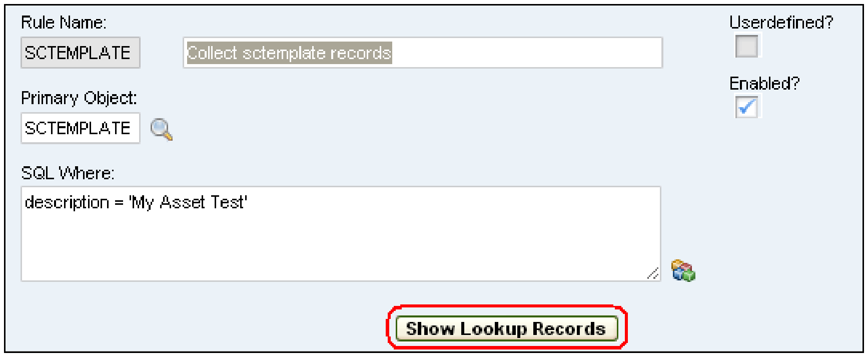
- DMSCTEMPLATE
- DMQUERY

8.4.5 Migration Collections definition

After the change requirements are reviewed and the object structures are identified, the next step is to define the migration collection contents that are used in the migration manager package.

Complete the following steps to create the collection definition:

1. Create a Migration Collection definition package.
2. In the lookup rules window, set the Where clause for the SCTEMPLATE rule. In our use case scenario, we used the start center template description in the Where clause. We can check the returned result by clicking **Show Lookup Records**, as shown in Figure 8-11.



Rule Name: SCTEMPLATE — Collect sctemplate records

Userdefined? ☐

Enabled? ✓

Primary Object: SCTEMPLATE

SQL Where:

```
description = 'My Asset Test'
```

Figure 8-11 Start center template lookup rule.

3. Add the QUERY object to the DMCOLLLOOKUP domain. This object can be used as a primary object in a lookup rule.
4. Add a lookup rule and enable it to migrate the FAILEDTEST query that is used in the start center result set. The rule should include the following parameters:
   - Rule Name: QUERY
   - Description: Collect queries
   - Primary Object: QUERY
   - SQL Where clause name: FAILEDTEST

Important: Steps 3 and 4 define a Lookup rule for queries and are, therefore, similar to the lookup rule that is defined in 6.3.3, “Solution B: Migration collection” on page 153.

5. After we defined all the rules that we need for the Migration Collection definition, run a migration collection validation. This step automatically adds the lookup rules records to the package. To see the result, we must click **Refresh Collection Objects** or refetch the collection package.
6. Create the migration package definition.

8.4.6 Deployment

After the package definition is created, approved, and activated, you create and distribute the package to the target environment. In our use case scenario, we do not need to activate the Admin mode to deploy the package; we must upload and deploy only the package in the target environment.

8.4.7 Deployment considerations

When you create a start center template in a development environment, the Tivoli's process automation engine assigns an internal SCTEMPLATEID number to the start center template. This number is used to uniquely identify the template and can be used in the Security Groups application to assign the template to a specific security group.

When start center templates are migrated from one environment to another, the SCTEMPLATEID can be different in the target environment if the two systems are not a copy of each other. It is important to manually check the security groups in the target environment and update the start center template identifier if necessary.

9 Migrating workflows

Workflow represents a defined business process that is layered on a *Maximo business object* (MBO). It is used extensively to automate many aspects of the business. For example, you can use a workflow process to automate the approval routing of a user-submitted purchase requisition or change request.

In its simplest form, a workflow process can consist of a conditional action that is based on a submitted record criteria and requires no user interaction. More complex workflow processes can be split into subprocesses and are used to manage highly interactive change management approval processes, such as multiple assignments, actions (status updates), and notifications via email or pager systems by using predefined communication templates.

This chapter includes the following sections:

- Requirements
- Configuration applications
- Solution A: Snapshot migration
- Solution B: Collection migration
- Deployment
- Deployment considerations
- Migrating a workflow with a circular dependency

9.1 Requirements

A workflow is often defined in a development environment and is promoted to test production environments. A repeatable process is required to reliably (without any errors or failures) and accurately (without missing any required content) migrate the workflow processes.

In this scenario, we describe the migration of a workflow that contains subprocesses, actions, roles, and communication templates. We demonstrate how to include the subprocesses and all of the workflow components in the same migration package by using two different approaches. For the purposes of this demonstration, we use the ISMACCEPT workflow, which is available in the SmartCloud Control Desk 7.5 demonstration database.

First, the ISMACCEPT workflow and all of its components are migrated to a target environment by using a snapshot migration package. We create a migration group and identify the SQL selection clauses that are required to move the workflow. Second, the same workflow is migrated; however, this time by using a migration collection. We demonstrate how to add each configuration component to the collection and create a package for migration.

It is assumed that the underlying data dictionary and other dependency objects were migrated to the target environment before the workflow was migrated.

9.2 Configuration applications

The following configuration applications are used for this workflow:

- Roles

  To access the Roles application, click **Go To** → **System Configuration** → **Platform Configuration** → **Roles**.

- Actions

  To access the Actions application, click **Go To** → **System Configuration** → **Platform Configuration** → **Actions**.

- Communication templates

  To access the Communication Templates application, click **Go To** → **System Configuration** → **Platform Configuration** → **Communication Templates**.

- Workflows

  To access the Workflow application, click **Go To** → **System Configuration** → **Platform Configuration** → **Workflow Designer**.

9.3 Solution A: Snapshot migration

After the migration requirements for this case scenario are reviewed, we identified an approach so that all the related configuration content is migrated by using a single snapshot migration package. The package is configured to use the standard migration group for business process management (BPM), which is installed by the Tivoli's process automation engine.

From a content perspective, the migration of the workflow and its related content requires the migration of the following configuration objects:

- Actions
- Action Groups
- Roles
- Communication templates
- Workflow processes and subprocesses

Important: Configuration objects that require migration for workflows are not limited to the list of objects that are described here. There are many instances in which other objects might need to be migrated, so it is important to consider workflow dependencies. Keep in mind the following considerations when you are deciding which objects to migrate:

- Actions that are used in the workflow can rely upon the existence of an application action or custom database attribute.
- Roles can depend upon the existence of people, person groups, or a custom attribute that is related to the record.
- Communication templates can depend upon roles, people, and person groups.
- Interaction nodes in workflow can reference a custom application, a custom select action menu item, a custom relationship, or another workflow process.
- Expressions that are used in workflow process nodes and actions can require the existence of domains, custom database attributes, classifications, and relationships.

All dependencies must be captured in the migration process.

9.3.1 Object structures

The following standard object structures that are provided with the product are used to support the migration requirements:

- DMACTION
- DMACTIONGROUP
- DMROLE
- DMCOMMTEMPLATE
- DMWFPROCESS

9.3.2 Migration groups

After the object structures are identified, the next step is to define the migration groups.

The standard group BPM (which ships with the product) contains the required migration objects. We create a group that is based on this group for our scenario.

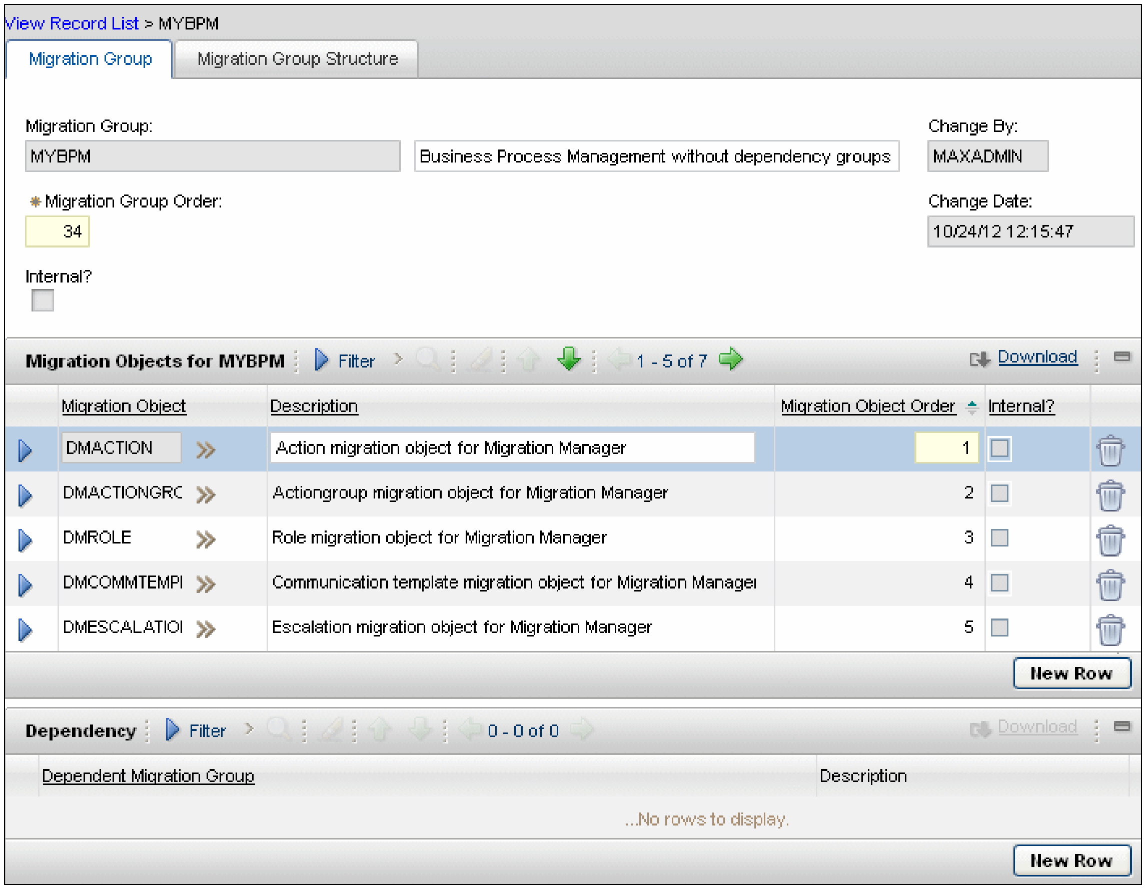
Create the group by duplicating the standard BPM group first, leaving all of the defaults, and then complete the following steps:

1. Name the group `MYBPM` with the description `Business Process Modules without dependencies`.
2. Remove all dependency groups.
3. Save the group.

Retaining the existing objects in the duplicated BPM group allows for future use of the new group to migrate other similar configurations; for example, escalations.

By removing the dependency groups, we are excluding the configuration content of these dependent groups. However, dependencies always must be part of the migration process. In this use case scenario, we work on an existing workflow. Therefore, we assume that the configuration on which the workflow elements might depend is already in the target environment. For example, if a table attribute was created or a new person group for a role was added, these objects must be migrated before you attempt to migrate the workflow.

Figure 9-1 shows the new migration group.



View Record List > MYBPM

Migration Group | Migration Group Structure

Migration Group: MYBPM | Business Process Management without dependency groups

Change By: MAXADMIN

Migration Group Order: 34

Change Date: 10/24/12 12:15:47

Internal? ☐

Migration Objects for MYBPM Filter 1 - 5 of 7 Download

Migration Object	Description	Migration Object Order	Internal?
DMACTION	Action migration object for Migration Manager	1	☐
DMACTIONGRC	Actiongroup migration object for Migration Manager	2	☐
DMROLE	Role migration object for Migration Manager	3	☐
DMCOMMTEMPI	Communication template migration object for Migration Managei	4	☐
DMESCALATIOI	Escalation migration object for Migration Manager	5	☐

New Row

Dependency Filter 0 - 0 of 0 Download

Dependent Migration Group	Description

...No rows to display.

New Row

Figure 9-1 New migration group MYBPM

9.3.3 Package definitions

In this solution, we create and deploy one snapshot migration package that contains the workflow process and its related configurations.

Create a package definition and name it WFISMACCEPTMM. Set the package type to snapshot and add the previously created migration group MYBPM.

The next step is to define the SQL selection criteria for every object structure in the migration group. The SQL queries can be devised to require only the name of the main workflow process, ISMACCEPT. The WHERE clauses for the related workflow components, such as roles, actions, and communication templates, use in-line SQL statements that are built to retrieve the information within each object by using the name of the main workflow process.

Important: SQL statements can be simplified if the workflow developer uses a naming convention when the workflow and its associated objects are created. For example, devising an SQL statement to search for all actions where the name starts with ISM for the DMACTION migration object can be reduced to the following code: `action like 'ISM%'`.

Table 9-1 lists the SQL WHERE clauses for each object. These queries add one or more configuration objects to the migration package.

Table 9-1 WHERE clauses for the WFISMACCEPTMM migration package

MIgration object	Object	Where clause
DMACTION	ACTION	action in (select action from WFACTION where (processname, processrev) in (select processname, processrev from wfrevision where mainprocess = 'ISMACCEPT' and revision = (select max(revision) from wfrevision where mainprocess = 'ISMACCEPT'))) or action in (select member from actiongroup where action in (select action from WFACTION where (processname, processrev) in (select processname, processrev from wfrevision where mainprocess = 'ISMACCEPT' and revision = (select max(revision) from wfrevision where mainprocess = 'ISMACCEPT'))))

MIgration object	Object	Where clause
DMACTIONGROUP	ACTIONGROUP	action in (select action from WFACTION where (processname, processrev) in (select processname, processrev from wfrevision where mainprocess = 'ISMACCEPT' and revision = (select max(revision) from wfrevision where mainprocess = 'ISMACCEPT')))
DMROLE	MAXROLE	maxrole in (select sendtovalue from COMMTMPLTSENDTO where type='ROLE' and templateid in (select templateid from COMMTEMPLATE where templateid in (select templateid from WFNOTIFICATION where (processname, processrev) in (select processname, processrev from wfrevision where mainprocess = 'ISMACCEPT' and revision = (select max(revision) from wfrevision where mainprocess = 'ISMACCEPT')))))
DMCOMMTEMPLATE	COMMTEMPLATE	templateid in (select templateid from WFNOTIFICATION where (processname, processrev) in (select processname, processrev from wfrevision where mainprocess = 'ISMACCEPT' and revision = (select max(revision) from wfrevision where mainprocess = 'ISMACCEPT')))
DMESCALATION	ESCALATION	1=2
DMWFPROCESS	WFPROCESS	(processname = 'ISMACCEPT' and active = 1) or (processname <> 'ISMACCEPT' and (processname, processrev) in (select processname, processrev from wfrevision where mainprocess = 'ISMACCEPT' and revision = (select max(revision) from wfrevision where mainprocess = 'ISMACCEPT')))
DMINBOUNDCOMMCFG	INBOUNDCOMMCFG	1=2

The WHERE clauses for each object generate the precise content for this package. The queries with a logical false statement (1=2) generate no results because escalation and inbound communication content is not migrated in this case.

9.4 Solution B: Collection migration

In this solution, the workflow ISMACCEPT is migrated by using a collections approach. The collection is created by choosing the workflow processes and actions that are relevant to the migration and then running validation and selecting the relevant configuration objects from the related configurations that are returned from the validation process.

Collections offer various methods for collecting configuration content. We demonstrate a number of these methods within this solution. For more information about creating collections, see Chapter 3, “Migration collections” on page 49.

From a content perspective, the migration of the workflow ISMACCEPT requires the migration of the following configuration objects:

- Actions
- Action Groups
- Roles
- Communication templates
- Workflow processes and subprocesses

In general, the migration of workflow processes can rely on the migration of other configuration objects, as described in 9.3, “Solution A: Snapshot migration” on page 237.

9.4.1 Configuration for migration collections

To create the collection and perform the validation process on the collection, it is important that the system is configured to facilitate creating migration collections and to produce optimal validation results. The validation process can be tuned by setting the migration manager system properties and the Related Data and lookup rules. Migration collection support can be added to applications for easy selection of configuration objects within applications.

System properties

Set the system property `mxe.dm.collvalidlevels` to the value `2`. This setting limits the depth of the validation scan to two levels, which is the appropriate depth for retrieving records that are related to the workflow process.

For more information about this property and other system properties that control collection validation, see 3.4.3, “Managing validation rules” on page 58.

Related data rules

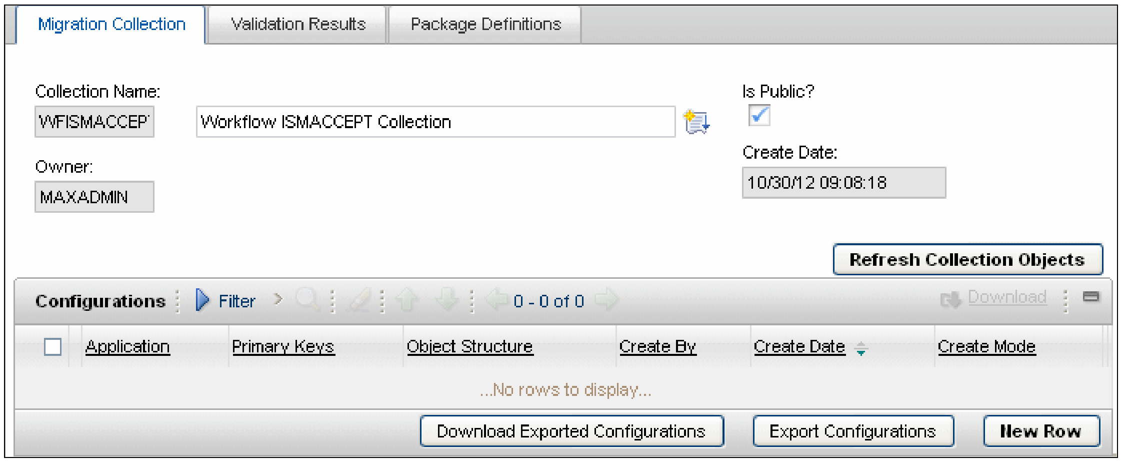
Enable the Related Data rule that is provided for ACTIONGROUP in the Migration Collections application by using the Define Related Data and Lookup Rules action. Disable all other Related Data rules to limit the results that are returned from the validation process.

Add migration collections to applications

Use the Add Migration Collections to Applications action in the Migration Collections application to enable migration collection application support for the Workflow Designer and Actions applications.

9.4.2 Creating the migration collection

Browse to **Go To** → **System Configurations** → **Migration** → **Migration Collections**. Create a collection and name it WFISMACCEPTCOL. Mark the collection as public so that other users can view and use the migration collection. This collection contains the workflow main process ISMACCEPT and all related subprocesses, actions, communication templates, and roles. Figure 9-2 shows the empty collection.



Collection Name: WFISMACCEP' Workflow ISMACCEPT Collection

Is Public? ✓

Owner: MAXADMIN

Create Date: 10/30/12 09:08:18

0 - 0 of 0

	Application	Primary Keys	Object Structure	Create By	Create Date	Create Mode

...No rows to display...

Figure 9-2 Migration collection WFISMACCEPTCOL

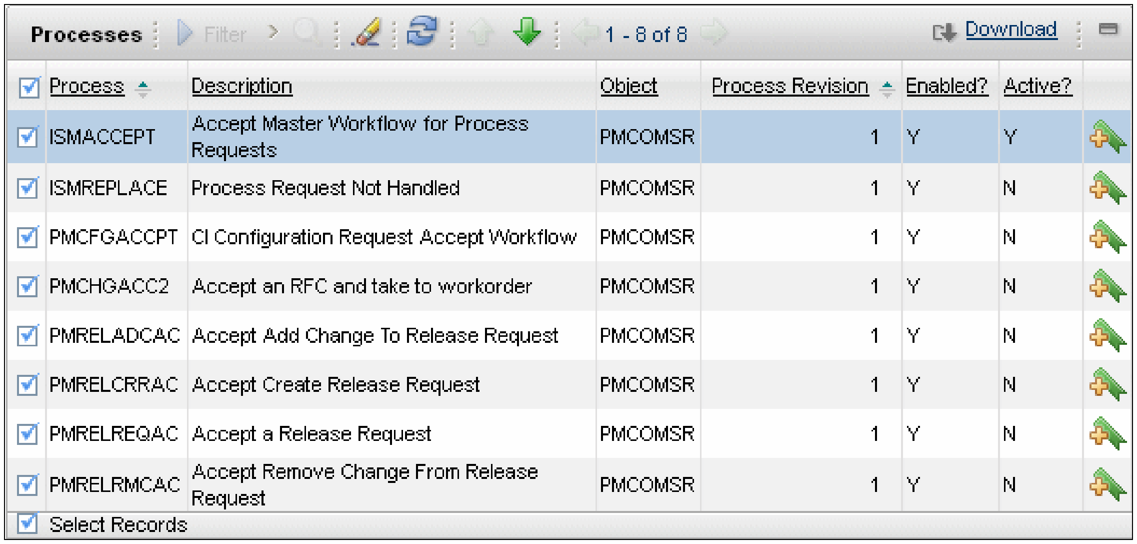
Workflow processes

Go to the Workflow Designer application by browsing to **Go To** → **System Configuration** → **Platform Configuration** → **Workflow Designer**. The Add To Migration Collections icon is now available on the toolbar because migration collections support was enabled for this application.

The standard search and filtering operations can be used to retrieve a list of the workflow processes to migrate. In this case, we presume that the user is aware of the workflow main process and subprocess names to migrate. Filter the list by searching for the wanted enabled processes by using the following process:

1. In the Process field, enter: `ISMACCEPT,PMCHGACC2,PMCFGACCPT,PMRELREQAC,PMRELADCAC,PMRELRMCAC,PMRELCRRAC,ISMREPLACE`.
2. In the Enabled field, enter: `Y`.

After the list is filtered, select all of the records in the list, as shown in Figure 9-3.



✓	Process	Description	Object	Process Revision	Enabled?	Active?
✓	ISMACCEPT	Accept Master Workflow for Process Requests	PMCOMSR	1	Y	Y
✓	ISMREPLACE	Process Request Not Handled	PMCOMSR	1	Y	N
✓	PMCFGACCPT	CI Configuration Request Accept Workflow	PMCOMSR	1	Y	N
✓	PMCHGACC2	Accept an RFC and take to workorder	PMCOMSR	1	Y	N
✓	PMRELADCAC	Accept Add Change To Release Request	PMCOMSR	1	Y	N
✓	PMRELCRRAC	Accept Create Release Request	PMCOMSR	1	Y	N
✓	PMRELREQAC	Accept a Release Request	PMCOMSR	1	Y	N
✓	PMRELRMCAC	Accept Remove Change From Release Request	PMCOMSR	1	Y	N

✓ Select Records

Figure 9-3 Workflow processes for migration

Click the **Add to Migration Collections** icon in the toolbar and select the collection that was created, WFISMACCEPTCOL. The workflow processes are added to the collection.

Tip: This step shows an important capability in Migration Collections to add multiple configurations into a collection from the List tab of the governing configuration application. This approach can save implementers significant time and effort when building collections.

Actions

Browse to the Actions application by selecting **Go To → System Configuration → Platform Configuration → Actions**. The Add To Migration Collections icon is now available on the toolbar because migration collections support was enabled for this application.

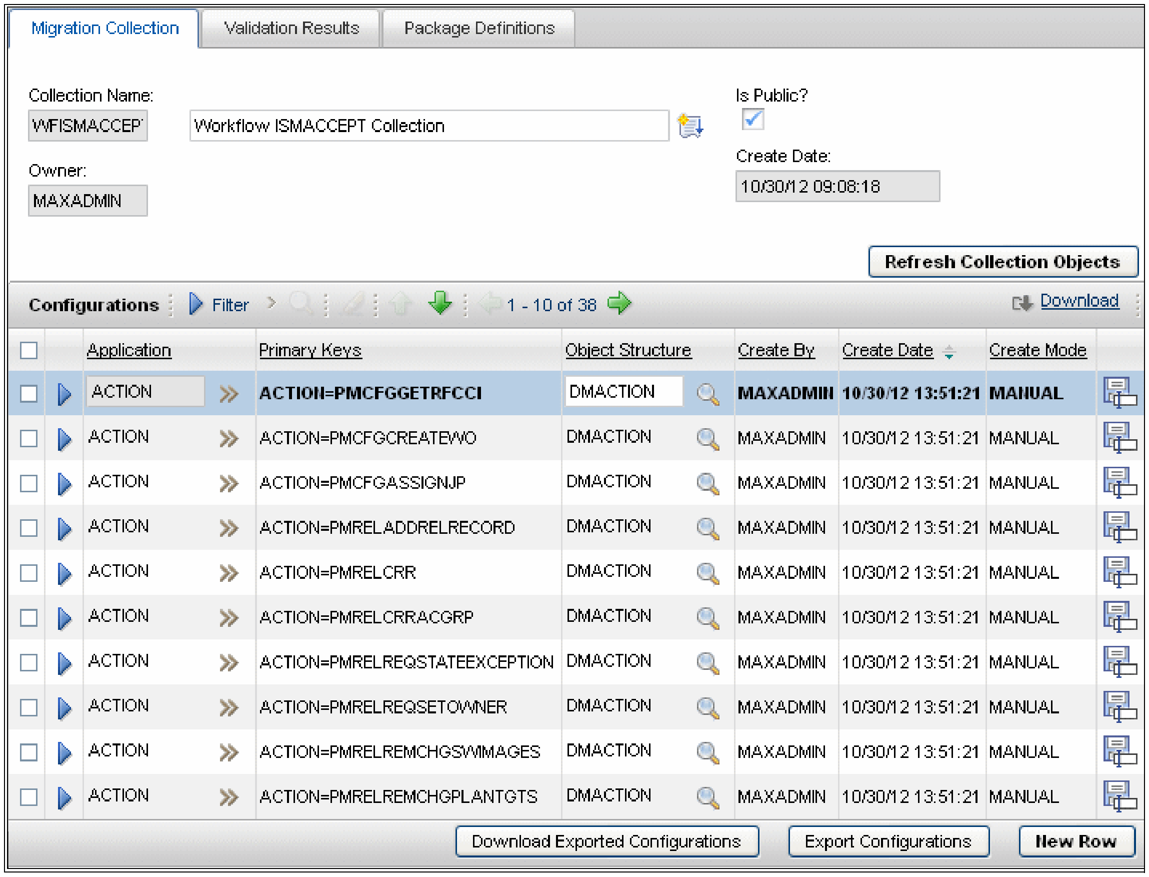
If a naming convention was in place for the actions that were created for the workflow process ISMACCEPT, the user can filter on the action name and object name by using the naming convention to retrieve the list of actions to be migrated. However, in this case it is assumed that a naming convention was not used. To obtain all actions that were used within the workflow process, enter the query that is shown in Example 9-1 in the **Advanced Search → Where Clause** window:

Example 9-1 Sample WHERE clause to retrieve actions

```
action in
   (select action
   from WFACTION
   where (processname, processrev) in
      (select processname, processrev
      from wfrevision
      where mainprocess = 'ISMACCEPT'
      and revision =
         (select max(revision)
         from wfrevision
         where mainprocess = 'ISMACCEPT')))
   or action in
      (select member
      from actiongroup
      where action in
         (select action
         from WFACTION
         where (processname, processrev) in
            (select processname, processrev
            from wfrevision
            where mainprocess = 'ISMACCEPT'
            and revision =
               (select max(revision)
               from wfrevision
               where mainprocess = 'ISMACCEPT'))))
```

The list returns all actions, including actions that belong to action groups as action members that are used in the most recently synchronized version of the workflow process ISMACCEPT and its subprocesses. Select all actions that are returned in the list and add them to the migration collection WFISMACCEPTCOL.

Figure 9-4 shows the migration collection WFISMACCEPTCOL after the workflow processes and actions are added.



Collection Name: WFISMACCEP Workflow ISMACCEPT Collection

Owner: MAXADMIN

Is Public? ✓

Create Date: 10/30/12 09:08:18

	Application	Primary Keys	Object Structure	Create By	Create Date	Create Mode
☐	ACTION	ACTION=PMCFGGETRFCCI	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMCFGCREATEWO	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMCFGASSIGNJP	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELADDRELRECORD	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELCRR	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELCRRACGRP	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELREQSTATEEXCEPTION	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELREQSETOWNER	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELREMCHGSWIMAGES	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL
☐	ACTION	ACTION=PMRELREMCHGPLANTGTS	DMACTION	MAXADMIN	10/30/12 13:51:21	MANUAL

Figure 9-4 WFISMACCEPTCOL with workflow processes and actions

Validation

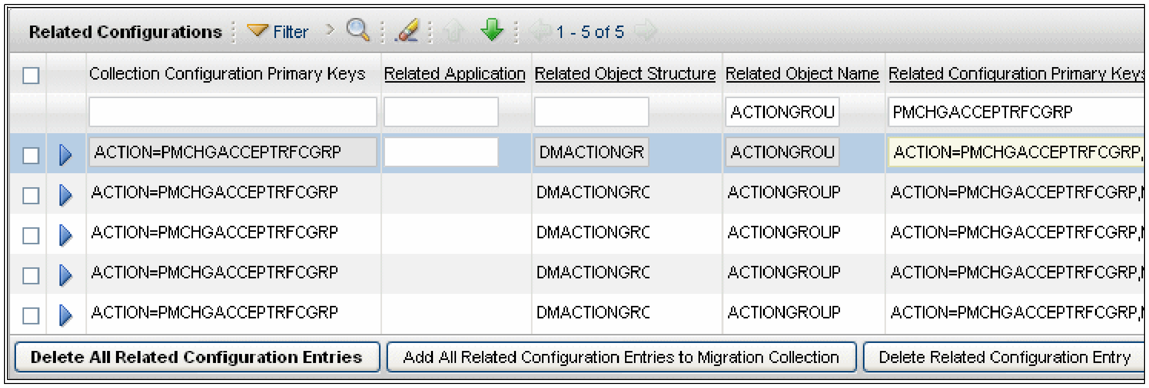
A successful migration of the workflow contains workflow processes, actions, action groups, communication templates, and roles within the migration package. Action groups, communication templates, and roles are added to the collection by using the validation process. In the Migration Collections application, use the Validate Migration Collection action to start the validation process.

After validation is successfully completed, click the **Validation Results** tab on the Migrations Collections application. The validation process returns numerous objects that relate to the migration collection. These results can be analyzed and then added to the migration collection or removed from the related configurations set.

The workflow process ISMACCEPT uses the following action groups:

- PMCHGACCEPTRFCGRP
- PMRELADCACGRP
- PMRELRMCACGRP
- PMRELCRRACGRP

To search for these related configurations, set the filter criteria on the Related Configurations subtable to filter where Related Object Name is ACTIONGROUP. Validation picked up action groups, which are used by the workflow process ISMACCEPT and action groups, which are not used by the workflow process. Filter the list further by adding to the search criteria the first action group name, PMCHGACCEPTRFCGRP, in the Related Configuration Primary Keys field. The resulting set, which contains all action members for the action group, is shown in Figure 9-5.



Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
			ACTIONGROU	PMCHGACCEPTRFCGRP
ACTION=PMCHGACCEPTRFCGRP		DMACTIONGR	ACTIONGROU	ACTION=PMCHGACCEPTRFCGRP,
ACTION=PMCHGACCEPTRFCGRP		DMACTIONGRC	ACTIONGROUP	ACTION=PMCHGACCEPTRFCGRP,I
ACTION=PMCHGACCEPTRFCGRP		DMACTIONGRC	ACTIONGROUP	ACTION=PMCHGACCEPTRFCGRP,I
ACTION=PMCHGACCEPTRFCGRP		DMACTIONGRC	ACTIONGROUP	ACTION=PMCHGACCEPTRFCGRP,I
ACTION=PMCHGACCEPTRFCGRP		DMACTIONGRC	ACTIONGROUP	ACTION=PMCHGACCEPTRFCGRP,I

Delete All Related Configuration Entries
Add All Related Configuration Entries to Migration Collection
Delete Related Configuration Entry

Figure 9-5 Filtering related configurations

Select all five records and click **Add Related Configuration Entry to Migration Collection**. Retrieve the records for the remaining three action groups (PMRELADCACGRP, PMRELRMCACGRP, PMRELCRRACGRP) and add them to the collection.

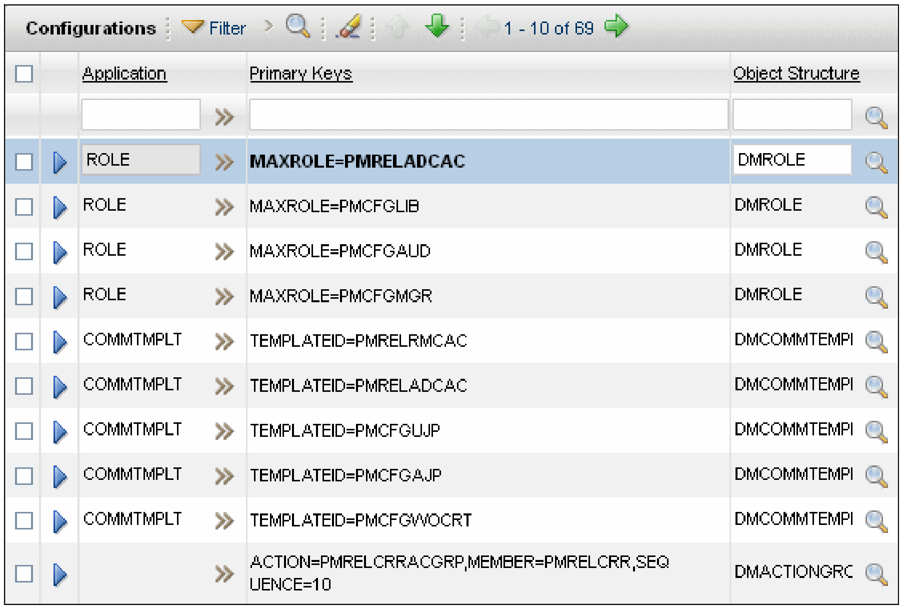
To add communication templates that are used within the workflow process to the collection, filter the Related Configurations where Related Object Name is COMMTEMPLATE. Add all returned results to the migration collection.

Finally, add the roles that are used in the workflow and communication templates by filtering the validation results where the Related Object Name is MAXROLE. Add all returned results to the collection.

Related roles: Roles that are related to communication templates were found by the validation process because the system property `mxe.dm.collvalidlevels` is set to a value of `2`.

No other migration objects are required to be migrated in this collection. Select **Delete All Related Configuration Entries**.

The final migration collection that contained all workflow processes, actions, action groups, communication templates, and roles is shown in Figure 9-6.



Application	Primary Keys	Object Structure
ROLE	MAXROLE=PMRELADCAC	DMROLE
ROLE	MAXROLE=PMCFGLIB	DMROLE
ROLE	MAXROLE=PMCFGAUD	DMROLE
ROLE	MAXROLE=PMCFGMGR	DMROLE
COMMTMPLT	TEMPLATEID=PMRELRMCAC	DMCOMMTEMPI
COMMTMPLT	TEMPLATEID=PMRELADCAC	DMCOMMTEMPI
COMMTMPLT	TEMPLATEID=PMCFGUJP	DMCOMMTEMPI
COMMTMPLT	TEMPLATEID=PMCFGAJP	DMCOMMTEMPI
COMMTMPLT	TEMPLATEID=PMCFGWOCRT	DMCOMMTEMPI
	ACTION=PMRELCRRACGRP,MEMBER=PMRELCRR,SEQUENCE=10	DMACTIONGRC

Figure 9-6 Finalized WFISMACCEPTCOL collection

Click the **Create Package Definition** action on the migration collection. The package definition that is created is viewable on Package Definitions tab in the Migration Collections application. Select the link to go to the Migration Manager application to create and distribute the migration manager package.

9.5 Deployment

After all package definitions are saved, approved, and activated, the next step is to create and distribute the package to the target environment. After the package is available to the target environment, it is time to deploy.

Load and deploy either the WFISMACCEPTMM package (Solution A) or the WFISMACCEPTCOL package (Solution B) in the target environment. The deployment does not require enabling Admin mode.

Although dependencies exist between the workflow main process and the subprocesses, the workflow main process and subprocesses can be deployed in a single migration package. The workflow processes are created as inactive and are disabled in the target environment.

9.6 Deployment considerations

Because you are updating an existing workflow, there might be transactions under this workflow in the target environment. The records that are affected continue to use the old version of the workflow.

The updated workflow is deployed as inactive and disabled, and its subprocesses are not enabled. To complete the deployment, you must enable the migrated subprocesses and then enable and activate the migrated main workflow process.

Version

It is common practice to develop and test multiple versions of a workflow process. However, you migrate the chosen active version, and the version resets to the next available version number in the target environment.

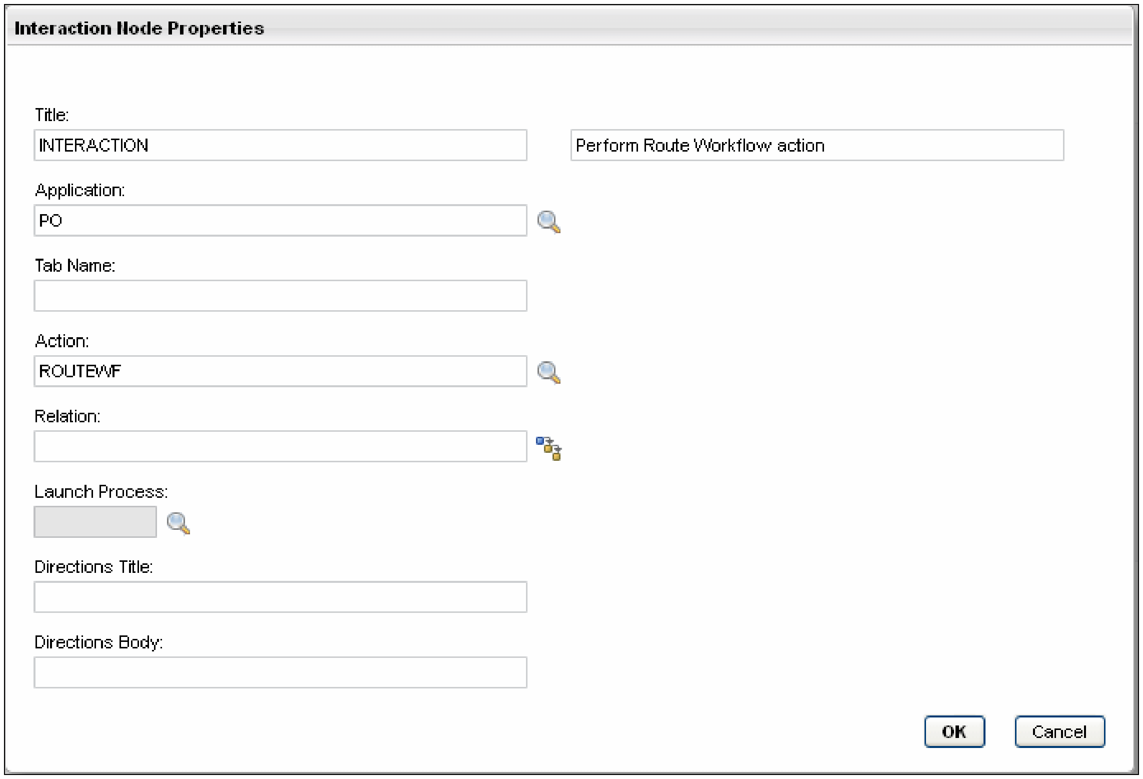
Activation

Although a manual activation step is necessary, this step also gives you the opportunity to verify the contents of the workflow and to activate only the workflow at the appropriate point in the target environment.

9.7 Migrating a workflow with a circular dependency

The first time a workflow is migrated, workflow support must be added to the application if the application does not already have workflow support that is enabled. Workflow support is added to the application in the target environment when the workflow is enabled and activated.

Workflow support can be added to the application at any time during workflow development by using the Add Workflow to Applications action in Workflow Designer. After workflow support is added to the application, workflow actions are available to the user in the application. One example of the workflow actions that is added is the Route Workflow action. These workflow actions are also made available to be selected in an interaction node within the workflow process, as shown in Figure 9-7.


Interaction Node Properties

Title: INTERACTION

Perform Route Workflow action

Application: PO

Tab Name:

Action: ROUTEWF

Relation:

Launch Process:

Directions Title:

Directions Body:

Figure 9-7 Interaction node that uses the Route Workflow action

You encounter a circular dependency migration case when you must migrate a workflow that contains a reference to a workflow action and you must add workflow application support in the target. The workflow cannot pass the migration validation because it contains a reference to workflow application actions that are not enabled in the target. However, workflow support cannot be added to the application because the workflow does not exist in the target.

One method of resolving this dependency is to create two migration packages. In the first migration package, remove all references to the workflow actions from the workflow. Migrate the workflow and add workflow application support in the target environment. In the source environment, restore the workflow actions in the workflow process. Create the second migration package and migrate the workflow to the target environment.

There is an alternative to this method in which you can migrate the workflow in a single migration package. First, create a migration collection that contains the workflow in the source environment. The workflow contains references to workflow actions and workflow support are added to the application in the source environment. When you run validation, the select action menu items for the workflow actions are returned in the related configurations. Select action menu items are stored in the table MAXMENU and use the DMMAXMENU object structure. There are eight menu options and one menu header that are created for the application when workflow support is added to the application. Filter for all nine maxmenu objects in the related configurations and add them to the collection. For example, the configurations for workflow actions in the PO application are shown in Table 9-2 on page 252.

Table 9-2 Workflow maxmenu configurations

Object Structure	**Primary Keys**
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =HEADER,KEYVALUE=AM10
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=SPECWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=LOGWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=HELPWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=VIEWWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=ASSIGNWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=HISTORYWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=STOPWF
DMMAXMENU	MENUTYPE=APPMENU,MODULEAPP=PO,ELEMENTTYPE =OPTION,KEYVALUE=ROUTEWF

The workflow migrates to the target environment in a single migration package. After the workflow is migrated, go to the Workflow Designer application and use the Add Workflow Support to Applications action to ensure that workflow support is added to the application in the target environment.

10

Migrating classifications

Business data grows daily. To ease reporting and understand the data better, you must classify data according to your business needs.

This chapter provides information about migrating classification data via Migration Manager, and includes the following sections:

- Scenario
- Configuration applications
- Solution A: Snapshot migration
- Solution B: Collection migration
- Deployment
- Deployment considerations

10.1 Scenario

Classifications are hierarchical data with which you can group and classify data. Users can report and understand the problem distribution easier with classification data. You can use classifications in many applications. Also, you can define separate attributes for each classification. Attributes are the characteristics of the data. Attributes help improve data consistency and make data search easier. For example, consider that there are two classifications. One classification is a service classification that is named NET.NETWORKSERVICE, and the other classification is an application server classification that is named APP.J2EE.WEBSPHERE.WEBSPHERESERVER. You can see the examples of the classification attributes in Table 10-1. These classifications are available in the SmartCloud Control Desk 7.5 demonstration database.

Table 10-1 Example of classification attributes

NET.NETWORKSERVICE classification attributes	APP.J2EE.WEBSPHERE.WEBSPHERE SERVER classification attributes
Service name	Status
Context IP	Product name
Bidirectional (BIDI) flag	Product version

You can search configuration items (CIs) via their attributes. For example, you can search CIs with an Active status and Version 6.0.2.1, according to your classifications.

There might be separate domains that are attached to these attributes to provide data consistency. You can refer to Chapter 5, “Migrating the data dictionary” on page 95 for more information.

You must create classifications to group data according to your grouping needs.

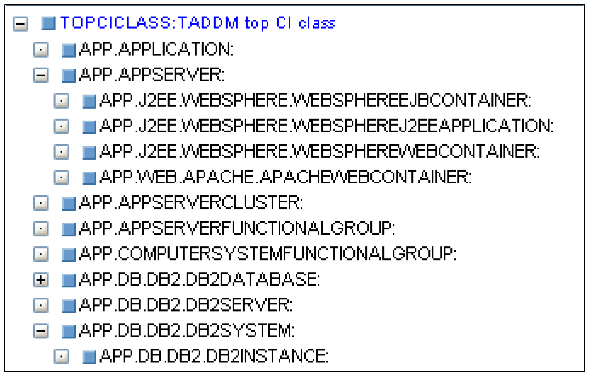
Classification data is hierarchical. Classifications can have a maximum of one parent, but they might have infinite child classifications. Classifications must be transferred from the development environment to the production environment sequentially. Classifications that are promoted must be migrated without an interruption.

Attributes and units of measure that are referenced by the classifications that are migrated must be migrated before the classifications.

10.2 Configuration applications

You can define, manage, and delete classifications in the Classifications application. The Classifications application is started by clicking **Go To** → **Administration** → **Classifications**.

Figure 10-1 shows an example of a classification hierarchy.



```
TOPCICLASS:TADDM top CI class
  APP.APPLICATION:
  APP.APPSERVER:
    APP.J2EE.WEBSPHERE.WEBSPHEREEJBCONTAINER:
    APP.J2EE.WEBSPHERE.WEBSPHEREJ2EEAPPLICATION:
    APP.J2EE.WEBSPHERE.WEBSPHEREWEBCONTAINER:
    APP.WEB.APACHE.APACHEWEBCONTAINER:
  APP.APPSERVERCLUSTER:
  APP.APPSERVERFUNCTIONALGROUP:
  APP.COMPUTERSYSTEMFUNCTIONALGROUP:
  APP.DB.DB2.DB2DATABASE:
  APP.DB.DB2.DB2SERVER:
  APP.DB.DB2.DB2SYSTEM:
    APP.DB.DB2.DB2INSTANCE:
```

Figure 10-1 Example of a classification hierarchy

Before classifications are migrated, be sure to migrate organizations, person groups, domains, measure units, and service groups if you used these objects in classification or attribute definitions.

Figure 10-2 shows the supporting objects.

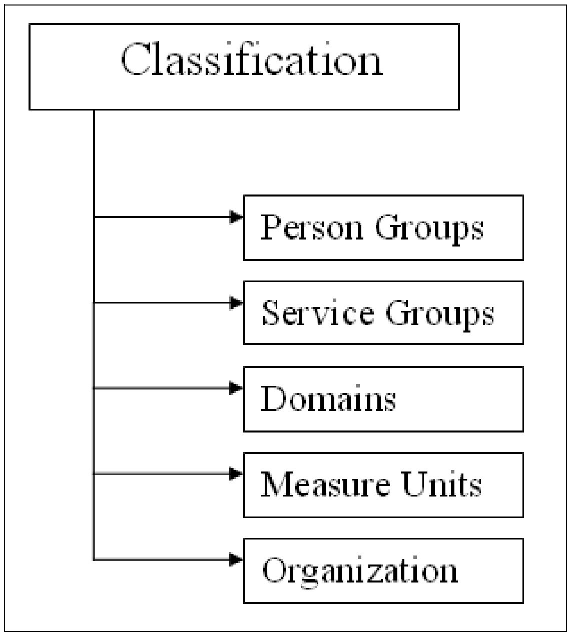


Figure 10-2 Supporting objects of classification data

10.3 Solution A: Snapshot migration

This section describes migrating classifications by using snapshot migration package definitions. You specify the set of classifications to migrate by creating a SQL WHERE clause that selects the classifications.

10.3.1 Object structure

You cannot use the predefined DMCLASSIFICATION object structure to migrate classification data because classification data has a child and parent hierarchy. Also, this object structure is not enough for a hierarchical data structure. Child data and parent data might be created in a random order in the source environment, which can lead to the random export order of classifications. If the child classification is exported before the parent classification, it causes an error because the migration process is unable to find the parent classification in the database while the child classification is processed.

To prevent this situation, create an object structure to migrate the data successfully. Make the new object structure recursive. It must begin from the top-level classification. When the Migration Manager processes the top level, it processes the next level until there is no lower level of classification.

You can see the predefined object structure named DMCLASSIFICATION in Figure 10-3.

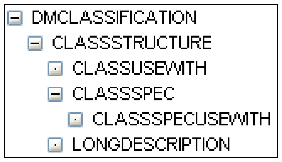


Figure 10-3 DMCLASSIFICATION object structure

To duplicate the object structure, browse to **Go To** → **System Configuration** → **Migration** → **Object Structures**.

Select the DMCLASSIFICATION object structure and then select Duplicate Object Structure in the Select Action menu, as shown in Figure 10-4.

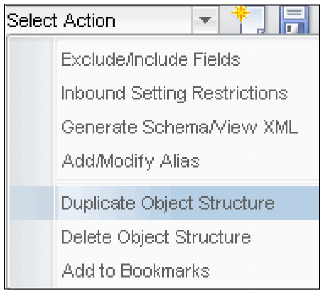
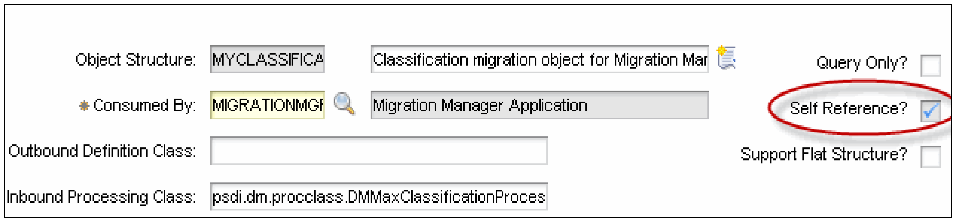


Figure 10-4 Duplicate Object Structure option in the Select Action menu

When you duplicate the object structure, give it a name, such as MYCLASSIFICATION, and then select the Self Reference option to make the object structure recursive, as shown in Figure 10-5.



Object Structure: MYCLASSIFICA Classification migration object for Migration Mar

Query Only? ☐

* Consumed By: MIGRATIONMGF Migration Manager Application

Self Reference? ✓

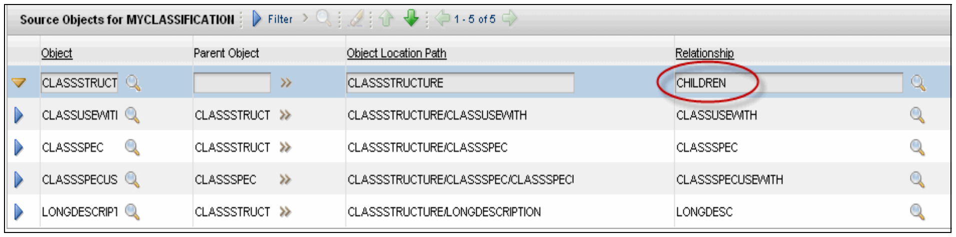
Outbound Definition Class:

Support Flat Structure? ☐

Inbound Processing Class: psdi.dm.procclass.DMMaxClassificationProces

Figure 10-5 Self Reference option in the object structure

When you select the Set Reference option, you see that the relationship field next to the first object in the object structure becomes editable. Enter `CHILDREN` in the relationship field that is next to the first object, as shown in Figure 10-6.



Source Objects for MYCLASSIFICATION

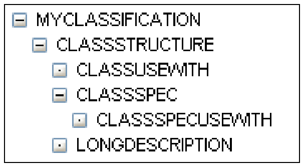
Object	Parent Object	Object Location Path	Relationship
CLASSSTRUCT		CLASSSTRUCTURE	CHILDREN
CLASSUSEWITI	CLASSSTRUCT	CLASSSTRUCTURE/CLASSUSEWITH	CLASSUSEWITH
CLASSSPEC	CLASSSTRUCT	CLASSSTRUCTURE/CLASSSPEC	CLASSSPEC
CLASSSPECUS	CLASSSPEC	CLASSSTRUCTURE/CLASSSPEC/CLASSSPECI	CLASSSPECUSEWITH
LONGDESCRIPT	CLASSSTRUCT	CLASSSTRUCTURE/LONGDESCRIPTION	LONGDESC

Figure 10-6 Relationship name

You now have the correct object structure to create a migration group.

Relationships: The child relationship is a relationship from the classstructure object to the same object. It is used to find the child records for a classstructure. The WHERE clause of the relationship is parent=:classstructureid.

Figure 10-7 shows the new object structure.



```
MYCLASSIFICATION
  CLASSSTRUCTURE
    CLASSUSEWITH
    CLASSSPEC
      CLASSSPECUSEWITH
    LONGDESCRIPTION
```

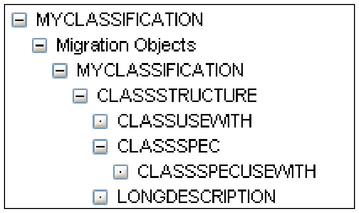
Figure 10-7 MYCLASSIFICATION object structure

10.3.2 Migration group

To migrate classification data, complete the following steps to create a migration group that includes the new object structure:

1. Browse to **Go To** → **System Configuration** → **Migration** → **Migration Groups**.
2. Create a migration group.
3. Give the group a name, such as MYCLASSIFICATION.
4. Add the MYCLASSIFICATION object structure to the migration group.

The migration group has only one migration object, as shown in Figure 10-8.



```
MYCLASSIFICATION
  Migration Objects
    MYCLASSIFICATION
      CLASSSTRUCTURE
        CLASSUSEWITH
        CLASSSPEC
          CLASSSPECUSEWITH
        LONGDESCRIPTION
```

Figure 10-8 Migration group of classification data

This migration group does not depend on other groups. You can directly migrate this group without any group order.

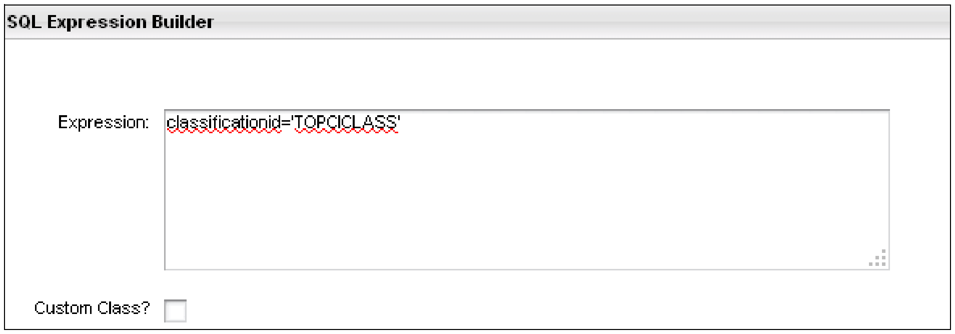
10.3.3 Package definition

To start the migration process, define the migration package. Create a single migration package and add the MYCLASSIFICATIONGROUP migration group to the package. The remainder of the package definition is described in the *Migration Manager Guide*, which is available at this website:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tamit.doc_7.1/pdf/mam71_migration_mgr_guide.pdf`

After the package is defined, create the SQL statement to define the classification data that you want to migrate.

Then, create a statement, similar to the statement that is shown in Figure 10-9.



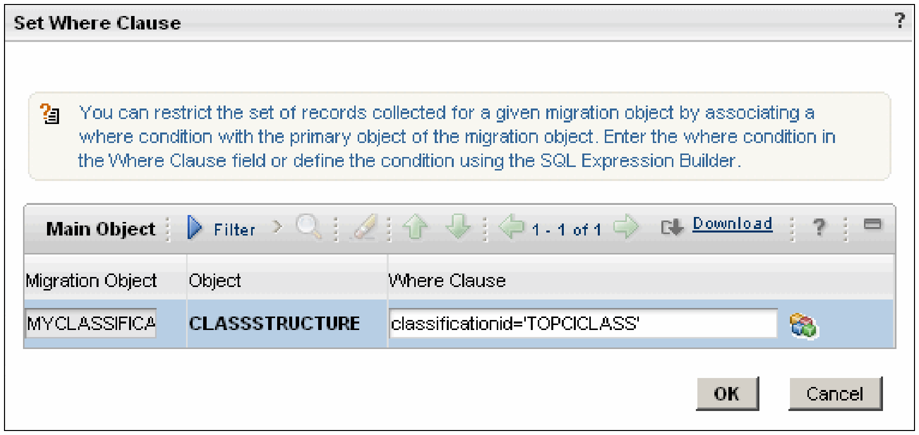
Expression: classificationid='TOPCICLASS'

Figure 10-9 Classification migration SQL statement

Tip: Migration Manager migrates all child classifications of any classifications that are returned by your SQL. You must select only the roots of any hierarchies that you want to migrate. You can restrict your selection to only top-level classifications by including “and parent is null” in your SQL.

The Migration Manager validates the SQL statement. There is an error message before the package is saved if any SQL statement fails on validation.

With this method, classification data is migrated from the top level to the bottom level consecutively. The Migration Manager gets all of the classifications recursively. You can see the WHERE clause statement window in Figure 10-10.


Set Where Clause

You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

Migration Object	Object	Where Clause
MYCLASSIFICA	CLASSSTRUCTURE	classificationid='TOPCICLASS'

Figure 10-10   Where clause window

Now, you can save and approve the package.

10.4 Solution B: Collection migration

This section describes migrating classifications by using migration collections. You specify the classifications to migrate by adding classifications to a migration collection.

10.4.1 Object structures

Classifications form a hierarchy. The upper levels of the hierarchy must be migrated before the lower levels. The predefined DMCLASSIFICATION object structure does not support hierarchical data. You must define a new object structure with hierarchy support. You make this definition most easily by duplicating the DMCLASSIFICATION object structure, and then make small changes to it. This procedure is described in 10.3.1, "Object structure" on page 257.

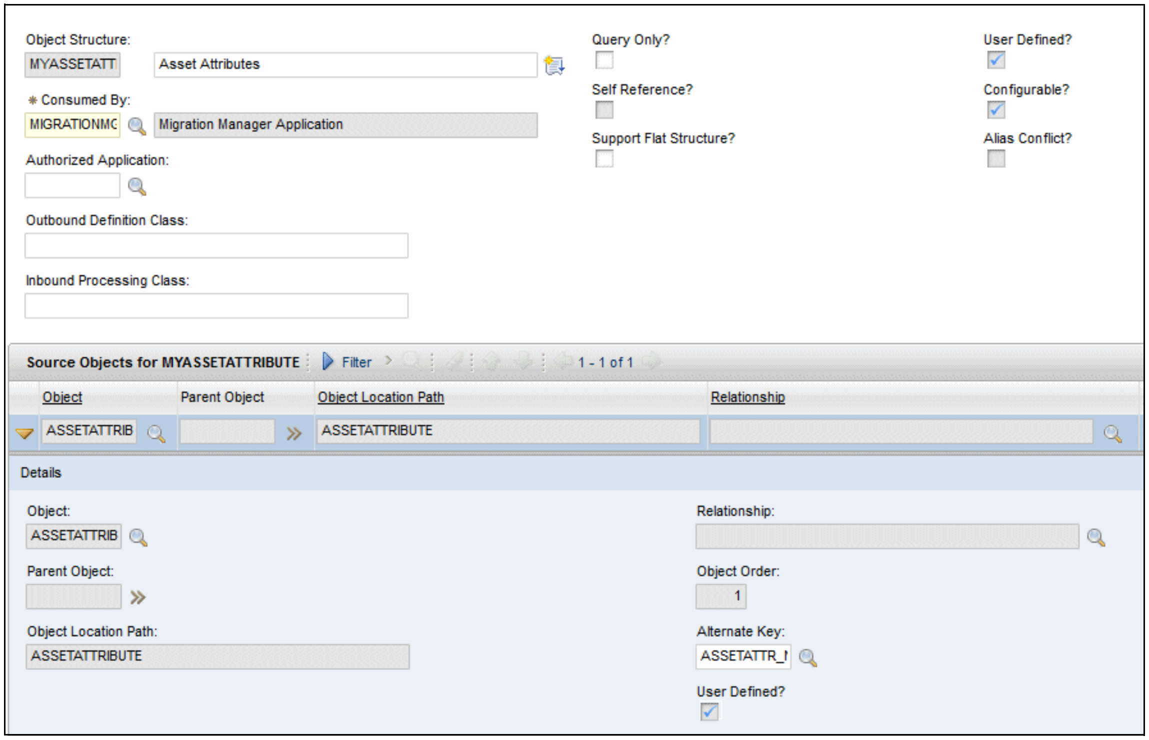
You must create an object structure for attributes if you created or modified any attributes that are referenced by the classifications you are migrating. To create the object structure, browse to **Go To** → **System Configuration** → **Migration** → **Object Structures**.

Give the new object structure a name, such as MYASSETATTRIBUTE. Select MIGRATIONMGR in the Consumed By field. Do not select the Self Reference option that you set on the classification object structure.

Add one record to the Source Objects list. Select ASSETATTRIBUTE as the object type. This object type is the correct type to use regardless of whether the attribute is used on asset classifications or some other type of classification.

Select ASSETATTR_NDX1 in the Alternate Key field. This field controls which attributes are used to search for existing ASSETATTRIBUTE records when migration deployments are done. The default index for this table includes only the ASSETATTRIBUTEID column. This value is not preserved when an attribute is migrated, so it cannot be used to determine whether a record in the migration package is a new attribute or an update to an existing attribute. The ASSETATTR_NDX1 includes the attribute's ORGID, SITEID, and ASSETATTRID columns. These values are preserved during the migration and they uniquely identify each attribute.

Figure 10-11 shows the final state of the Object Structure application after the MYASSETATTRIBUTE object structure is created.


Object Structure: MYASSETATT Asset Attributes

Query Only? ☐

User Defined? ✓

* Consumed By: MIGRATIONMG Migration Manager Application

Self Reference? ☐

Configurable? ✓

Authorized Application:

Support Flat Structure? ☐

Alias Conflict? ☐

Outbound Definition Class:

Inbound Processing Class:

Source Objects for MYASSETATTRIBUTE — Filter — 1 - 1 of 1

Object	Parent Object	Object Location Path	Relationship
ASSETATTRIB		ASSETATTRIBUTE	

Details

Object: ASSETATTRIB

Relationship:

Parent Object:

Object Order: 1

Object Location Path: ASSETATTRIBUTE

Alternate Key: ASSETATTR_I

User Defined? ✓

Figure 10-11 Creating the MYASSETATRIBUTE object structure

10.4.2 Migration group

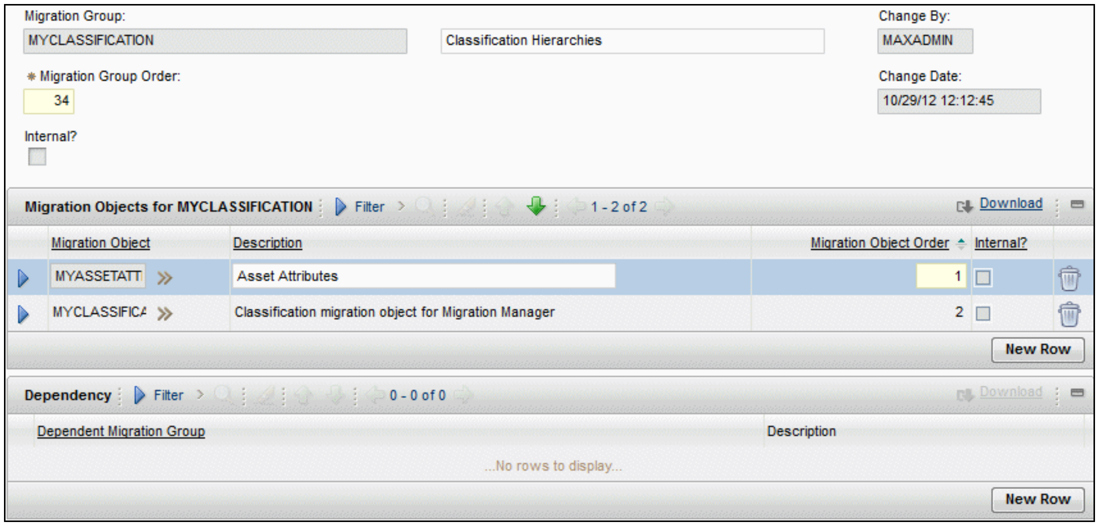
A migration collection cannot use your new object structures unless they are in a migration group. You must create a migration group to contain the MYCLASSIFICATION and MYASSETATTRIBUTE object structures you created.

Complete the following steps to create a migration group that includes the new object structures:

1. Browse to **Go To** → **System Configuration** → **Migration** → **Migration Groups**.
2. Create a migration group.
3. Give it a name, such as MYCLASSIFICATION.
4. Add the MYASSETATTRIBUTE object structure to the migration group.
5. Add the MYCLASSIFICATION object structure to the migration group.

This group is the only migration group that is used in the migration, so its Migration Group Order value is not significant. However, the Migration Object Order of the MYASSETATTRUBUTE object structure must be lower than that order of the MYCLASSIFICATION object structure. This order ensures that the attributes are deployed before any classifications that reference them.

Figure 10-12 shows the final state of the Migration Groups application after the MYCLASSSIFICATION migration group is saved.


Migration Group: MYCLASSIFICATION — Classification Hierarchies

Change By: MAXADMIN

* Migration Group Order: 34

Change Date: 10/29/12 12:12:45

Internal? ☐

Migration Objects for MYCLASSIFICATION — 1 - 2 of 2

Migration Object	Description	Migration Object Order	Internal?
MYASSETATT	Asset Attributes	1	☐
MYCLASSIFIC	Classification migration object for Migration Manager	2	☐

Dependency — 0 - 0 of 0

Dependent Migration Group	Description

...No rows to display...

Figure 10-12 Creating the MYCLASSIFICATION migration group

10.4.3 Migration collection

This section describes how to create the migration collection that is used to migrate attributes and the classifications that use them. We then describe how to automatically create a migration package definition that is based on the collection.

Complete the following steps to create a migration collection:

1. Browse to **Go To** → **System Configuration** → **Migration** → **Migration Collections**.
2. Create a migration collection.
3. Give it a name, such as CLASSIFICATIONS.
4. Select the **Is Public** option if you want other users to add objects to this collection.

5. Add classifications to the collection.

   Deciding on which objects to add to the collection is a more complex decision when you are migrating classifications. Classifications are hierarchical, and that you migrate every classification that is added to the collection and all of their child classifications.

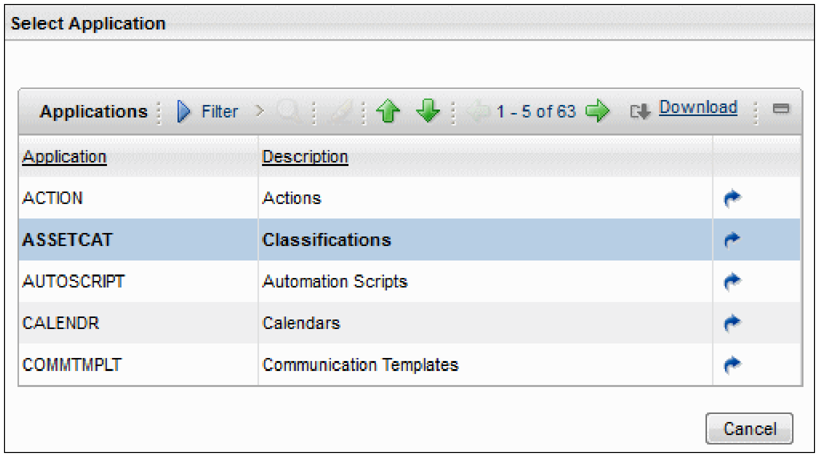
   Assume for this exercise that you changed the predefined classifications APP.J2EE.WEBSPHERE.WEBSPHERESERVER and NET.NETWORKSERVICE. Both of these classifications are a child of the ACTUALCIROOTCLASS classification.

   You could migrate the two classifications that you modified by adding the ACTUALCIROOTCLASS classification to the collection. However, ACTUALCIROOTCLASS has over 500 child classes that are be included in the migration. These classes might have other modifications that you are not ready to migrate. For this reason, we add the two modified classifications to the collection. This addition restricts the migration to just these two classifications and any of their children.

   Complete the following steps to add the classifications to the collection:

   a. Click **New Row** on the Migration Collection tab.

   b. Open the detail menu of the Application field to open the Application Selection panel, as shown in Figure 10-13.



Select Application

Applications | Filter | 1 - 5 of 63 | Download

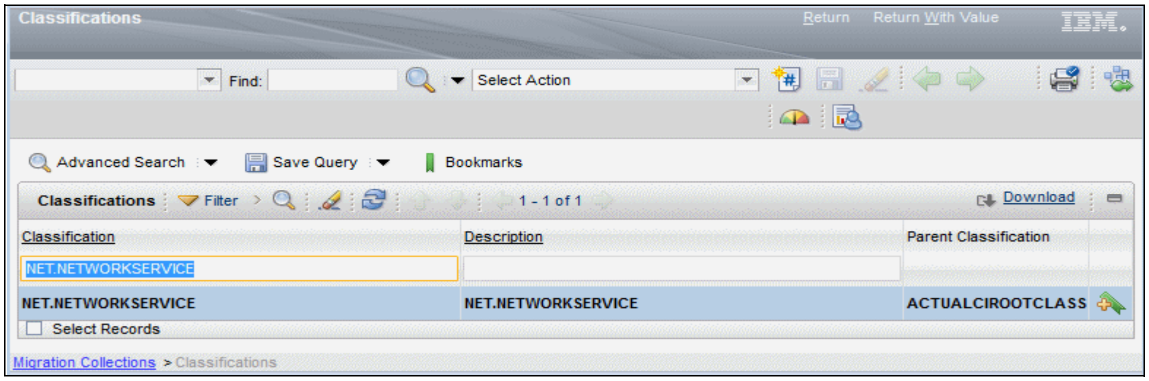
Application	Description
ACTION	Actions
ASSETCAT	Classifications
AUTOSCRIPT	Automation Scripts
CALENDR	Calendars
COMMTMPLT	Communication Templates

Cancel

Figure 10-13 Application selection

   c. Click the **Go To Application** icon that is to the right of the ASSETCAT line. The Classifications application opens.

d. Search for the NET.NETWORKSERVICE classification. Click the **Return With Value** link that is at the upper right of the window, as shown in Figure 10-14. The selected classification is added to the collection and you are returned to the Migration Collections application.



Classification	Description	Parent Classification
NET.NETWORKSERVICE		
NET.NETWORKSERVICE	NET.NETWORKSERVICE	ACTUALCIROOTCLASS

Figure 10-14 Selecting a classification to add to the migration collection

e. Verify that the added classification is using the wanted object structure.

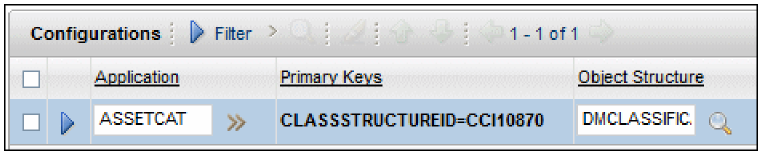
You created a MYCLASSIFICATION object structure to add support for migrating hierarchical data. Figure 10-15 on page 267 shows that the classification that you added to the collection is not using this object structure. Instead, it is using the predefined DMCLASSIFICATION object structure.

The Migration Collections application automatically selects an object structure when an object that is added to the collection is the primary object type of more than one object structure. This configuration is used in this example where classifications are the primary object for the DMCLASSIFICATION and MYCLASSIFICATION object structures.

When a choice must be made between multiple object structures, the choice is made based on the following criteria:

- Priority is given to object structures that are used in migration groups that are marked as “internal”.
- If multiple object structures still exist, priority is given to the structure that is used in the migration group with the lowest migration group order.

Change the object structure of the classification you added to the collection to MYCLASSIFICATION.



Application	Primary Keys	Object Structure
ASSETCAT	CLASSSTRUCTUREID=CCI10870	DMCLASSIFIC.

Figure 10-15 Classification added with wrong object structure

f. Repeat steps a - e to add the APP.J2EE.WEBSPHERE.WEBSPHERESERVER classification to the collection.

6. Add the MYASSETATTRIBUTE and MYCLASSIFICATION object structures to the collection.

   This step is necessary only for the first migration after these object structures are created. They must be included in the first deployment to any target environment. This inclusion ensures that the object structures are deployed and created before they are needed during the deployment of any attributes and classifications.

   Add the object structures to the collection in the same way that you added the classifications. Select the INTOBJECT application instead of the ASSETCAT application. You can accept the default object structure that is applied to these collection members.

7. Create a lookup rule that adds any new or modified attributes to the collection.

   Attributes cannot be added to the collection directly because they are not the primary object of any application that could be used to select them. By using lookup rules, you can add objects to collections by using SQL queries. This process is similar to the way you specify which objects to migrate in a snapshot-based migration package definition.

   Complete the following steps to create a lookup rule:

   a. Choose **Define Related Data** and **Lookup Rules** from the Select Action menu.

   b. Go to the Lookup Rules tab, and click **New Row**.

   c. Give the rule a name, such as ASSETATTRIBUTE.

   d. Select the primary object for the rule. For this exercise, select ASSETATTRIBUTE.

   e. Enter an SQL WHERE clause that selects records of the primary object type. For this exercise, there is one new attribute that was added to the APP.J2EE.WEBSPHERE.WEBSPHERESERVER classification. This one attribute can be selected by using the following WHERE clause:

      ```
      assetattrid = 'APPSERVER_SECURITYREALM'
      ```

      If you wanted to select every attribute that is used by any classification in the hierarchies that are rooted at the two classifications in the collection, you could use the following SQL WHERE clause:

      ```
      exists(select 1 from classspec where classspec.assetattrid =
      assetattribute.assetattrid and classspec.classstructureid in
      (select classstructureid from classancestor where ancestor in
      ('CCI10669','CCI10870')))
      ```

      The values `CCI10669` and `CCI10870` are the CLASSSTRUCTUREID values of the two classifications that are added to the collection. You can obtain these values from the Primary Keys column in the Configurations list on the Migration Collection tab. This WHERE clause assumes that the ASSETATTRID field of the attribute (which contains the attribute name) is enough to uniquely identify the attribute. This condition might not be true. Attributes with the same name can be defined in different organizations and sites. If this configuration is the case for you, you must extend the SQL in this example to compare the ORGID and SITEID columns of the ASSETATTRIBUTE and CLASSSPEC tables.

   f. Select the **Enabled** option.

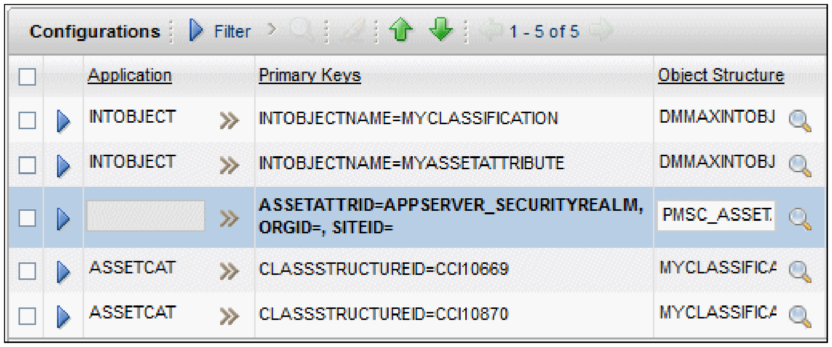
   g. Click **OK** to close the window.

8. Save the collection definition.
9. Select **Validate Migration Collection** from the Select Action menu, or click the icon on the toolbar.
10. Click **Start** in the Validate window. Periodically click **Refresh Status** until you see that the validation is complete. The validation process evaluates any lookup rules that are enabled and adds any objects that are selected by those rules to the collection.
11. Close the **Validate** window. You are brought automatically to the Validation Results tab of the Migration Collections application.

    Classifications can have dependencies on domains, persons, person groups, and service groups. The validation does not find any of these related objects during its normal scan because we configured the classifications in the collection to use an object structure that is not in an internal migration group. It is possible to define Related Data rules that find objects that are related to classifications, but this definition was not done in this example. For this reason, the Related Configurations list on the Validation Results tab is empty.

12. Return to the Migration Collection tab.

13. Click **Refresh Collection Objects**. You should see that the collection now contains the attributes that were returned by your lookup rule, as shown in Figure 10-16.



Configurations | Filter | 1 - 5 of 5

Application	Primary Keys	Object Structure
INTOBJECT	INTOBJECTNAME=MYCLASSIFICATION	DMMAXINTOBJ
INTOBJECT	INTOBJECTNAME=MYASSETATTRIBUTE	DMMAXINTOBJ
	ASSETATTRID=APPSERVER_SECURITYREALM, ORGID=, SITEID=	PMSC_ASSET.
ASSETCAT	CLASSSTRUCTUREID=CCI10669	MYCLASSIFICA
ASSETCAT	CLASSSTRUCTUREID=CCI10870	MYCLASSIFICA

Figure 10-16   Collection with lookup rule results

14. As was the case with the classifications, the object structure for the attribute is not the wanted object structure. Change the object structure of the attribute to MYASSETATTRIBUTE.

    You might see the correct object structure that is selected in your environment. The PMSC_ASSETATTRIBUTE object structure that was the default selection in this example is installed with an optional product extension. Not all installations have it.

    It is important to not use the PMSC_ASSETATTRIBUTE object structure in this case. The use of this object structure results in a migration package where each classification is included twice: once by using the DMCLASSIFICATION object structure, and once by using the MYCLASSIFICATION object structure.

    This usage happens because the PMSC_ASSETATTRIBUTE object structure belongs to the PMSC75_OFFERING migration group. This group also contains the DMCLASSIFICATION object structure. A migration package definition that was created from the collection references the migration groups that contain the object structures that are listed for all of the objects in the collection. In this case, the package definition references the PMSC75_OFFERING and MYCLASSIFICATION migration groups.

When a package is created based on the package definition, Migration Manager exports any objects in the collection by using any applicable object structure in its migration groups. In this case, it processes the PMSC75_OFFERING group first because of its lower migration group order. Any classifications in the collection are exported by using the DMCLASSIFICATION object structure because of its membership in the PMSC75_OFFERING group. The classifications are exported again by using the MYCLASSIFICATION object structure when the MYCLASSIFICATION migration group is processed.

Changing the attribute's object structure to MYASSETATTRIBUTE avoids this problem because it is a member of the MYCLASSIFICATION migration group only. None of the object structures that are referenced by members of the collection now are members of groups that include other object structures that are compatible with collection objects. This configuration prevents any double exports of objects.

15. Save the collection.

16. Select **Create Package Definition** from the Select Action menu. After the package definition is created, you can approve it and use it to create migration packages.

10.5 Deployment

After the package is created, you must distribute the package to get it to the target environment. You can distribute the package via file type distribution or database-type distribution. For more information, refer to Chapter 15, “Common topics” on page 351.

To migrate the data successfully, migrate the MYCLASSIFICATION object structure to the target environment. This migration is done automatically if you are using a collection-based package definition, and include any new object structures in the collection.

Classification data can have an organization, site, or person as the owner group and a service group that is attached to itself. Migrate these groups before migrating classification data. Otherwise, you can have validation errors while deploying classifications.

Tip: When you migrate classifications, you do not need to switch on Admin mode or restart the server.

After the deployment to the target environment, users can start to use the promoted classifications in the new environment.

10.6 Deployment considerations

To migrate new classifications successfully to the target environment, you and the users must never create classifications manually in the target environment. All of the data must be created first in the source environment and then migrated to the target environment. Otherwise, you can have index problems when the classification data is migrated to the target environment.

Snapshot migration: You must always use Snapshot or Collection migration for classification migration. The Migration Manager cannot handle the hierarchical structure when classifications are migrated with the Change package.

Before deployment, be sure that the object structure includes the HIERARCHYPATH non-persistent field in the CLASSSTRUCTURE object. Otherwise, you get the `BMXAA5592E - Inbound data does not have a valid hierarchy path` error. The Migration Manager relies on this attribute to determine whether inserts or updates must be performed.

The Migration Manager does not check whether the processing action is Replace or AddChange in a classification migration when the Snapshot migration is used. If the Migration Manager finds the same HIERARCHYPATH in the target environment, it updates the record.

11

Integration Framework migration

The Integration Framework provides predefined content and a toolkit for integrating application data with other applications. The Integration Framework is a flexible and configurable feature of the Tivoli's process automation engine. It offers various transport protocols, such as HTTP, web services, interface tables, and files. Data loading and export is performed based on the Maximo Business Object (MBO).

In data exchange scenarios, the Integration Framework is used to synchronize master and transactional data with external systems. For example, an integration between SmartCloud Control Desk and an external ticketing system could be created to synchronize tickets that were created in the external system with SmartCloud Control Desk by using the Integration Framework. Another example of an integration is an integration that is built between SmartCloud Control Desk and an external financial system, in which financial transactions that are generated in SmartCloud Control Desk must be sent to the external system.

Integration development is commonly performed in a development environment. The configuration is moved to a test, stage, and production environment. Migration Manager provides a reliable and efficient mechanism to transfer integration configurations from environment to environment.

This chapter includes the following sections:

- Requirements
- Configuration applications
- Solution
- Deployment
- Deployment considerations

11.1 Requirements

In this use case, we create an integration between Tivoli's process automation engine and an external system. The integration is based on a procurement scenario that uses purchase requisitions and purchase orders. We create the integration by using the predefined integration content and by configuring the system with the Tivoli's process automation engine integration applications.

We also demonstrate how to package the development in a migration collection and how to adjust the collection validation process to best suit our needs. Additionally, we see how replacement rules can be used to facilitate package migration when a different value for a configuration is required in the source and target environments. We deploy the migration package to the target environment without the obligation of system downtime.

The integration is created based on the following requirements:

- Purchase requisitions are created and approved in Tivoli's process automation engine. After a purchase requisition is approved in Tivoli's process automation engine, it is sent to the external system. The external system does not accept updates to the purchase requisition other than closures. The external system should be notified when the purchase requisition is closed.
- Purchase orders are created in the external system by using the purchase requisitions that are received from Tivoli's process automation engine. When a purchase order is created and approved in the external system, it is sent to Tivoli's process automation engine.
- The external system deploys a web service to receive purchase requisitions and status changes to purchase requisitions. Transactions should be received near-real time.
- The external system sends purchase orders to a synchronous web service that is deployed by Tivoli's process automation engine.

The integration is created and tested in a development environment and then migrated to a test environment. The external system also includes a development and test environment, and their web service is deployed in each environment.

11.2 Configuration applications

This section outlines the configuration for the integration. We create a purchase requisition object structure by using the predefined integration content. A new publish channel is created for the outbound purchase requisition transactions. The publish channel is configured to be event-based and to export events only when the purchase requisition’s status is changed to approved or closed. An endpoint for the external system’s purchase requisition web service also is created.

We create a purchase order object structure that is based on the system-provided purchase order object structure. An enterprise service is created by using this object structure for inbound transactions. A web service is deployed to the product container that is based on this enterprise service.

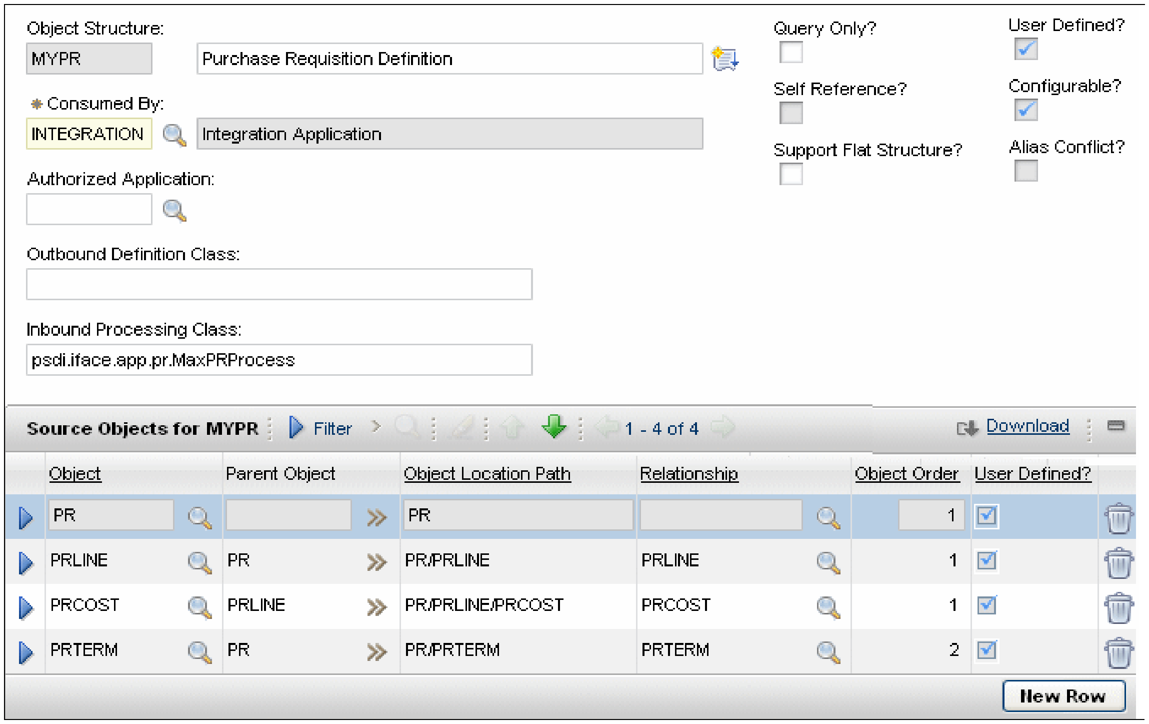
The publish channel and the enterprise service are added to a new external system, which represents the external system with which we are integrating. The integration configurations are described in this section.

11.2.1 Object structures

Complete the following steps to create the integration object structures by duplicating two object structures, which are provided with the product, MXPR and MXPO:

1. Click **Go To** → **Integration** → **Object Structures**.
2. Search for the object structure MXPR. Duplicate this object structure and name the copy MYPR.

   The new object structure is shown in Figure 11-1.



Object Structure: MYPR — Purchase Requisition Definition

* Consumed By: INTEGRATION — Integration Application

Authorized Application:

Outbound Definition Class:

Inbound Processing Class: psdi.iface.app.pr.MaxPRProcess

Query Only? ☐ | User Defined? ☑

Self Reference? ☐ | Configurable? ☑

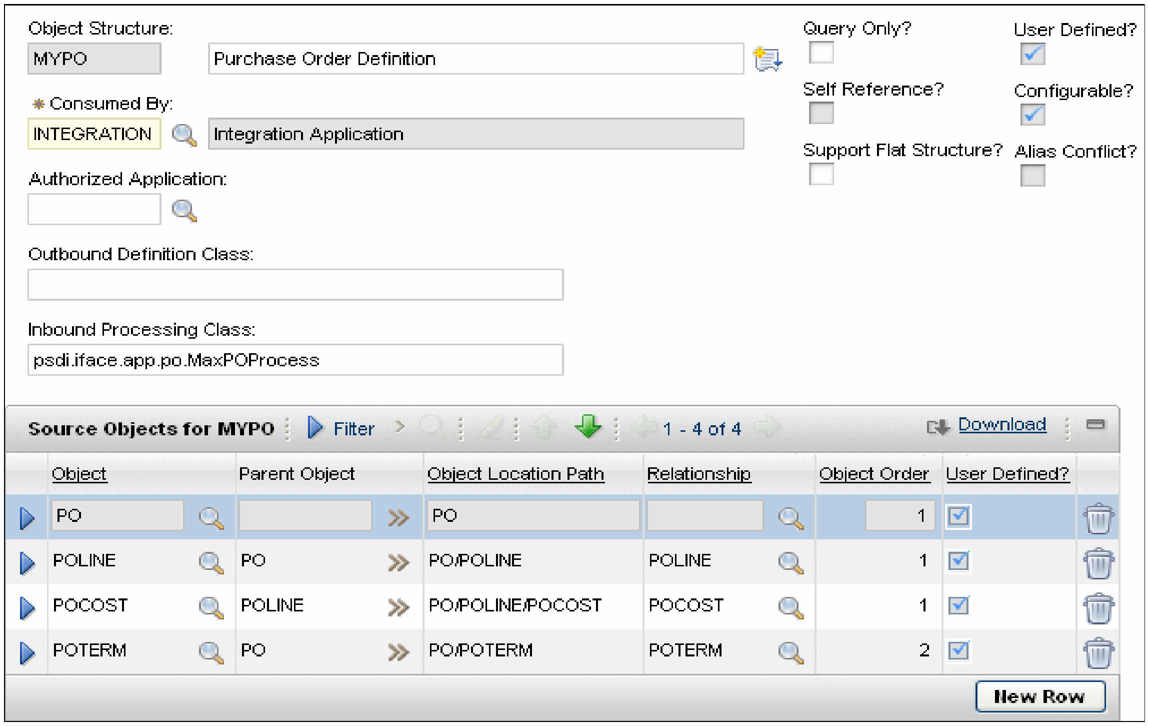
Support Flat Structure? ☐ | Alias Conflict? ☐

Source Objects for MYPR — Filter — 1 - 4 of 4 — Download

Object	Parent Object	Object Location Path	Relationship	Object Order	User Defined?
PR		PR		1	☑
PRLINE	PR	PR/PRLINE	PRLINE	1	☑
PRCOST	PRLINE	PR/PRLINE/PRCOST	PRCOST	1	☑
PRTERM	PR	PR/PRTERM	PRTERM	2	☑

Figure 11-1 MYPR object structure

3. Similarly, duplicate the object structure for MXPO and name it MYPO, as shown in Figure 11-2.



Object Structure: MYPO — Purchase Order Definition

Consumed By: INTEGRATION — Integration Application

Authorized Application:

Outbound Definition Class:

Inbound Processing Class: psdi.iface.app.po.MaxPOProcess

Query Only? ☐ User Defined? ✓

Self Reference? ☐ Configurable? ✓

Support Flat Structure? ☐ Alias Conflict? ☐

Source Objects for MYPO — 1 - 4 of 4

Object	Parent Object	Object Location Path	Relationship	Object Order	User Defined?
PO		PO		1	✓
POLINE	PO	PO/POLINE	POLINE	1	✓
POCOST	POLINE	PO/POLINE/POCOST	POCOST	1	✓
POTERM	PO	PO/POTERM	POTERM	2	✓

Figure 11-2 MYPO object structure

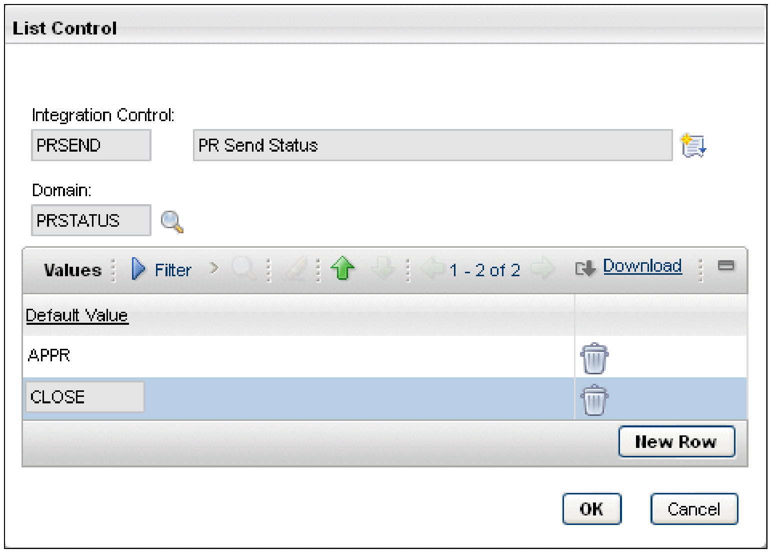
11.2.2 Publish channels

Complete the following steps to publish channels:

1. Click **Go To** → **Integration** → **Publish Channels**.
2. Search for MXPRInterface. Duplicate this publish channel and name it MYPRInterface.
3. Change the object structure from MXPR to the object structure that was created in 11.2.3, “Enterprise services” on page 282.

   In this integration, we send purchase requisitions to the external system when purchase requisitions have their status changed to approved or closed. All other updates to purchase requisitions are skipped. This control logic is established by creating two processing rules on the publish channel and modifying the integration control, which is already associated with the publish channel, PRSEND.

4. By using the Create Integration Controls action, open the PRSEND integration control. The APPR value exists in the list control.
5. Add a new value to the list control for the value CLOSE, as shown in Figure 11-3.



Integration Control: PRSEND PR Send Status

Domain: PRSTATUS

Default Value
APPR
CLOSE

Figure 11-3 PRSEND integration control

6. Create the first processing rule for the publish channel to skip all status changes except the statuses that are contained within the PRSEND integration control. The configuration for the processing rule is outlined in Table 11-1.

Table 11-1 MYPRInterface processing rule SKIPPR

Field	**Value**
Rule	SKIPPR
Description	Skip Outbound PR Transaction
Action	SKIP
Sequence	1
Apply on Primary Object Insert?	Y
Apply on Primary Object Update?	Y
Apply on Primary Object Delete	Y
Enabled	Y

The XML field condition that is associated with this processing rule is shown in Table 11-2.

Table 11-2 MYPRInterface processing rule SKIPPR XML field condition

Field	Evaluation Type	Evaluate When	Integration Control
STATUS	NOTEQUALS	ALWAYS	PRSEND

The object field condition that is associated with this processing rule is shown in Table 11-3.

Table 11-3 MYPRInterface processing rule SKIPPR object field condition

Object	Object Relationship	Field	Evaluate Type	Integration Control
PRSTATUS	PRSTATUS	STATUS	NOTEQUALS	PRSEND

Create the second processing rule to skip all other updates, as shown in Table 11-4.

Table 11-4 MYPRInterface processing rule SKIPPRUPDATE

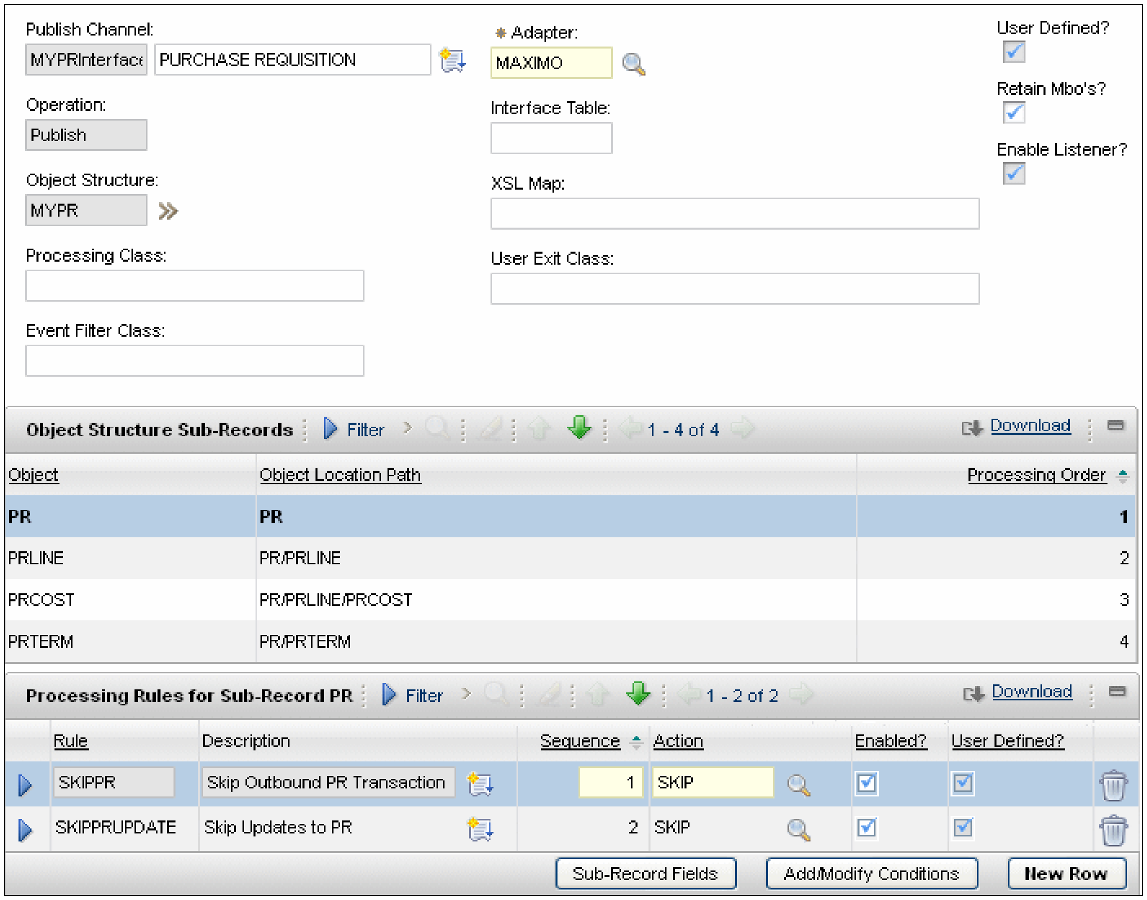
Field	Value
Rule	SKIPPRUPDATE
Description	Skip Updates to PR
Action	SKIP
Sequence	2
Apply on Primary Object Insert?	Y
Apply on Primary Object Update?	Y
Apply on Primary Object Delete	Y
Enabled	Y

The XML field condition for this rule is described in Table 11-5.

Table 11-5 MYPRInterface processing rule SKIPPRUPDATE XML field condition

Field	Evaluation Type	Evaluate When	Value
STATUSIFACE	EQUALS	ALWAYS	0

7. Enable the event listener by selecting the **Enable Event Listener** action. The publish channel appears, as shown in Figure 11-4.



Publish Channel: MYPRInterface PURCHASE REQUISITION

Adapter: MAXIMO

User Defined? ✓

Retain Mbo's? ✓

Enable Listener? ✓

Operation: Publish

Interface Table:

Object Structure: MYPR

XSL Map:

Processing Class:

User Exit Class:

Event Filter Class:

Object Structure Sub-Records Filter 1 - 4 of 4 Download

Object	Object Location Path	Processing Order
PR	**PR**	**1**
PRLINE	PR/PRLINE	2
PRCOST	PR/PRLINE/PRCOST	3
PRTERM	PR/PRTERM	4

Processing Rules for Sub-Record PR Filter 1 - 2 of 2 Download

Rule	Description	Sequence	Action	Enabled?	User Defined?
SKIPPR	Skip Outbound PR Transaction	1	SKIP	✓	✓
SKIPPRUPDATE	Skip Updates to PR	2	SKIP	✓	✓

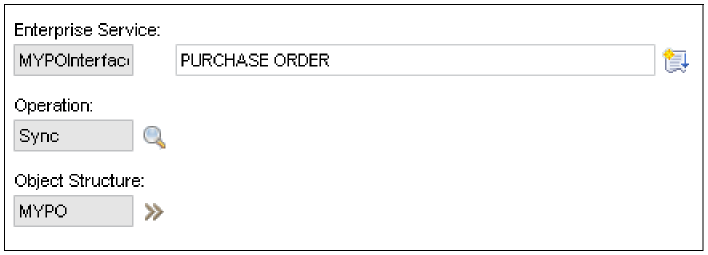
Sub-Record Fields | Add/Modify Conditions | New Row

Figure 11-4 MYPRInterface publish channel

11.2.3 Enterprise services

Complete the following steps to configure enterprise services:

1. Click **Go To** → **Integration** → **Enterprise Services**.
2. Duplicate the MXPOInterface and name it MYPOInterface.
3. Change the object structure to MYPO, as shown in Figure 11-5.



Enterprise Service: MYPOInterfac PURCHASE ORDER

Operation: Sync

Object Structure: MYPO

Figure 11-5 MYPOInterface enterprise service

11.2.4 Endpoints

Purchase requisitions are integrated to a web service that is hosted by the external system. In this scenario, the source environment is a Tivoli's process automation engine development environment. The external system also hosts a development environment on a server called *extdev*.

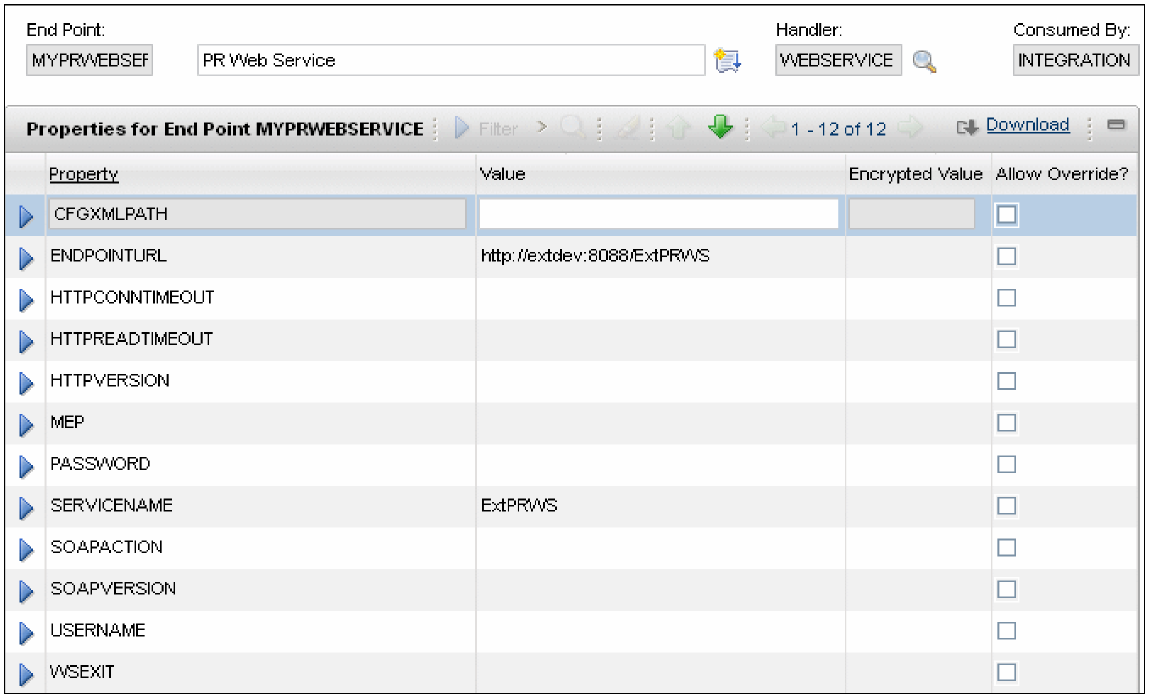
Complete the following steps to create an endpoint for the web service and set the endpoint's properties to connect to the external system's web service on their development server:

1. Click **Go To** → **Integration** → **End Points**.
2. Enter the values that are shown in Table 11-6.

Table 11-6 MYPRWEBSERVICE End Point

Field	**Value**
End Point	MYPRWEBSERVICE
Description	PR Web Service
Handler	WEBSERVICE
ENDPOINTURL	http://extdev:8088/ExtPRWS
SERVICENAME	ExtPRWS

The resulting endpoint is shown in Figure 11-6.



End Point: MYPRWEBSEF PR Web Service

Handler: WEBSERVICE

Consumed By: INTEGRATION

Properties for End Point MYPRWEBSERVICE Filter 1 - 12 of 12 Download

Property	Value	Encrypted Value	Allow Override?
CFGXMLPATH			☐
ENDPOINTURL	http://extdev:8088/ExtPRWS		☐
HTTPCONNTIMEOUT			☐
HTTPREADTIMEOUT			☐
HTTPVERSION			☐
MEP			☐
PASSWORD			☐
SERVICENAME	ExtPRWS		☐
SOAPACTION			☐
SOAPVERSION			☐
USERNAME			☐
WSEXIT			☐

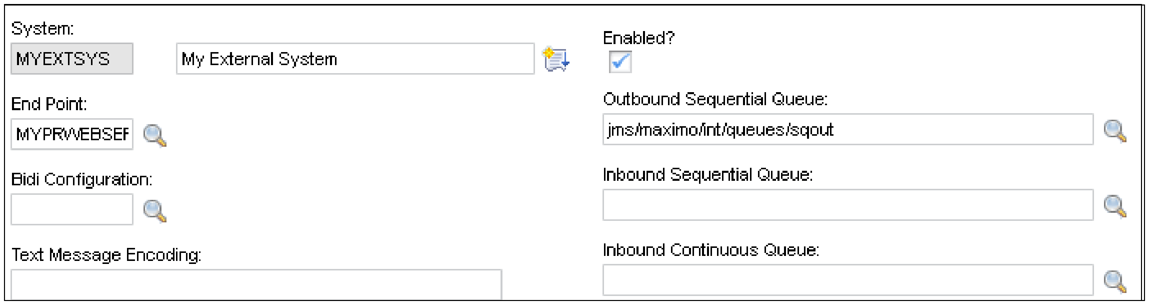
Figure 11-6 MYPRWEBSERVICE endpoint

11.2.5 External systems

Complete the following steps to configuration external systems:

1. Click **Go To** → **Integration** → **External Systems**.
2. Create an external system and name it MYEXTSYS.
3. Set the endpoint to the web service endpoint that was previously created, MYPRWEBSERVICE.
4. Select the system-provided outbound sequential queue sqout.

   The external system is shown in Figure 11-7.



System: MYEXTSYS My External System
End Point: MYPRWEBSEF
Bidi Configuration:
Text Message Encoding:
Enabled? ✓
Outbound Sequential Queue: jms/maximo/int/queues/sqout
Inbound Sequential Queue:
Inbound Continuous Queue:

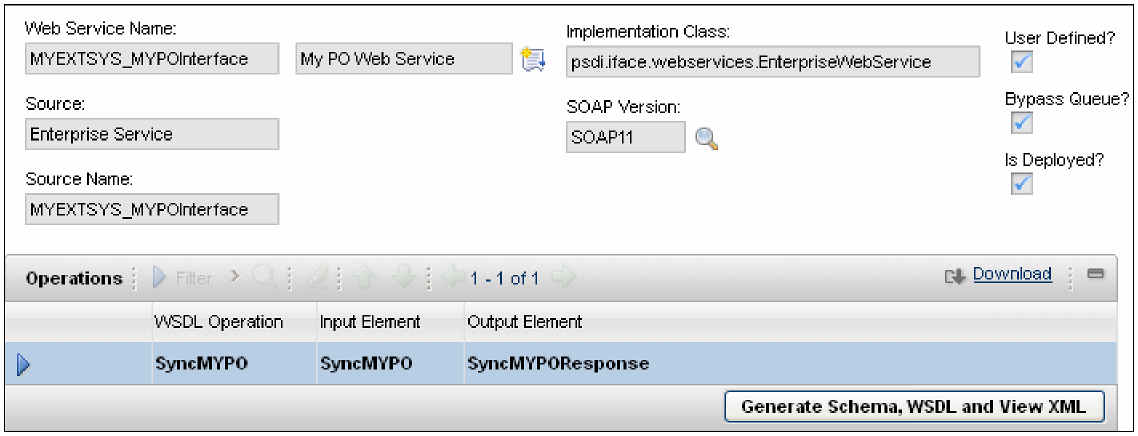
Figure 11-7 MYEXTSYS external system

5. Add the publish channel MYPRInterface and the enterprise service MYPOInterface.
6. The enterprise service bypasses the queues, so clear the **Use Continuous Queue?** option.
7. Enable the enterprise service, publish channel, and external system.

11.2.6 Web services library

Complete the following steps to configure the web services library:

1. Click **Go To** → **Integration** → **Web Services Library**.
2. Click **Create Web Service** → **Create Web Service from Enterprise Service**.
3. Select the enterprise service that was previously created, MYEXTSYS_MYPOInterface, and ensure that the web service is configured to bypass the queue.
4. Click **Create**.
5. Deploy the web service to the product container by selecting **Deploy to Product Web Service Container** → **Deploy Web Service**. The system property `mxe.int.containerdeploy` on the source environment and the target environment should be set to `0` to deploy web services to the product container. The web service is shown in Figure 11-8.



Web Service Name: MYEXTSYS_MYPOInterface | My PO Web Service

Source: Enterprise Service

Source Name: MYEXTSYS_MYPOInterface

Implementation Class: psdi.iface.webservices.EnterpriseWebService

SOAP Version: SOAP11

User Defined? ✓

Bypass Queue? ✓

Is Deployed? ✓

Operations

WSDL Operation	Input Element	Output Element
SyncMYPO	SyncMYPO	SyncMYPOResponse

Figure 11-8 MYEXTSYS_MYPOInterface web service

11.3 Solution

We migrate the integration development by using a collections approach. Migration Collections is a user-friendly application that we can use to collect the integration components. Instead of the use of complex SQL queries to specify the data that is contained in the package, we can use the Migration Collection application with the applications we used in 11.2, “Configuration applications” on page 276 to select each object we want to migrate. The result is a complete list of the configurations that are migrated. The collection can then be packaged and deployed onto the target environment.

11.3.1 Configuration for migration collections

In this section, we describe the configuration process that is used for migration collections.

System properties

In this example, we add all of the required objects for the migration to the collection, directly or with the use of lookup rules. We assume that any changes to objects or attributes that are used by the integration previously were migrated to the target system. With this assumption, we can decrease the validation process run time by configuring the migration collection system properties, as shown in Table 11-7. We also minimize package creation time by setting the system property `mxe.dm.autoapprovepkgdef` to automatically approve Migration Manager package definitions.

Table 11-7 System properties for MYINTCOL migration collection

System Property	Value
mxe.dm.collvalidlevels	1
mxe.dm.collvalidsrcexclude	<MAXINTOBJCOLS><MAXINTOBJALIAS><AUTOSCRIPT.OWNERID><AUTOSCRIPT.ORGID><AUTOSCRIPT.CREATEDBYNAME><AUTOSCRIPT.LOGLEVEL><AUTOSCRIPT.LANGCODE>
mxe.dm.collvalidtgtexclude	MAXINTOBJECT.USEWITH,MAXRELATIONSHIP.CARDINALITY,CRONTASKINSTANCE.SCHEDULE,AUTOSCRIPT.OWNERID,AUTOSCRIPT.ORGID,AUTOSCRIPT.CREATEDBYNAME,AUTOSCRIPT.LOGLEVEL,AUTOSCRIPT.LANGCODE,MAXLOGGER.LOGLEVEL,MAXATTRIBUTECFG,MAXOBJECTCFG
mxe.dm.autoapprovepkgdef	1

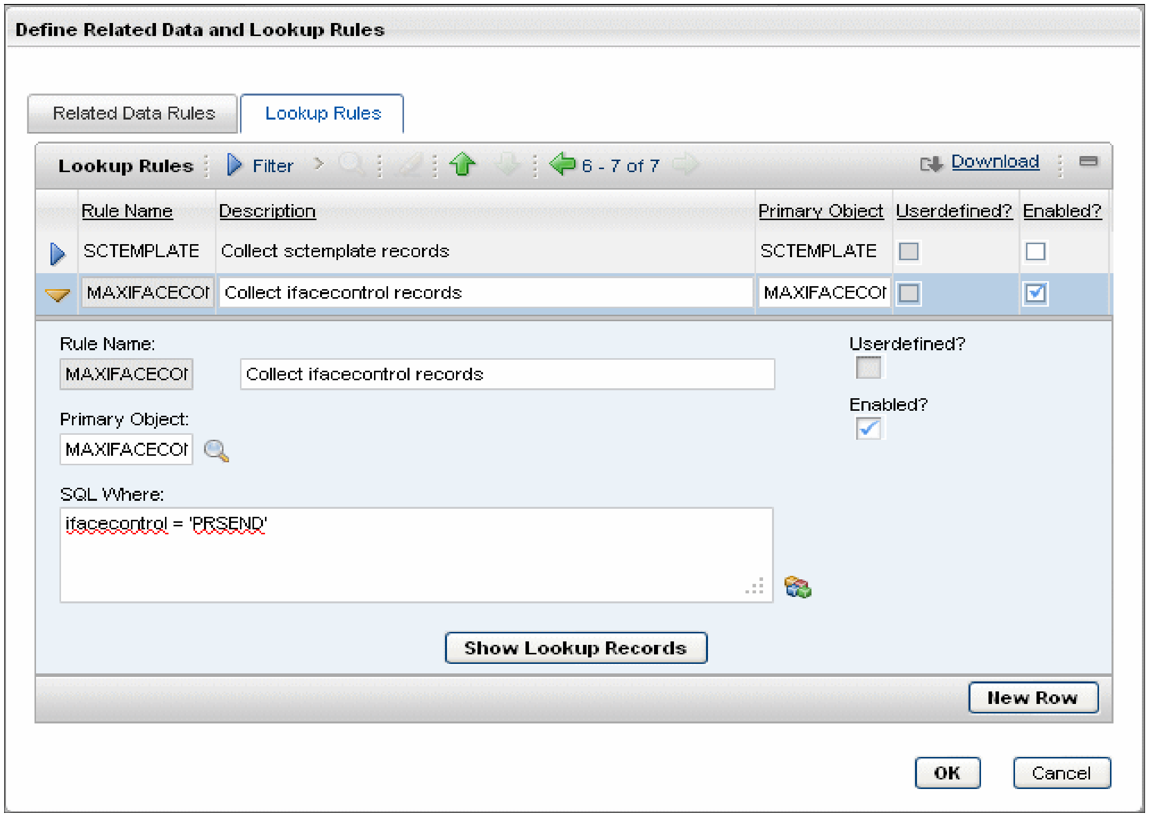
The removal of objects MAXATTRIBUTECFG and MAXOBJECTCFG from the validation scan can decrease the amount of time that is required to run the validation for the collection we are creating in this chapter. Conversely, these objects could produce valuable validation results to other developers who are creating migration collections in the same environment. These system properties apply to all migration collections that are created in the system. Therefore, their values should be agreed upon by all developers who are using the system.

Lookup rules

Integration controls are accessible within the Publish Channels and Enterprise Services applications. Because there is no dedicated application for integration controls, we cannot select the PRSEND integration control and add it to a collection explicitly. However, we can create a lookup rule for adding the PRSEND integration control to the migration collection.

Complete the following steps to create a lookup rule for adding the PRSEND integration control to the migration collection:

1. In the Migration Collections application, access the Lookup Rules window by selecting the **Define Related Data and Lookup Rules** action.
2. Modify the existing MAXIFACECONTROL lookup rule to select the PRSEND integration control, as shown in Figure 11-9. The SQL can be created by entering the WHERE clause manually or by using the SQL Expression Builder.



Rule Name	Description	Primary Object	Userdefined?	Enabled?
SCTEMPLATE	Collect sctemplate records	SCTEMPLATE	☐	☐
MAXIFACECOI	Collect ifacecontrol records	MAXIFACECOI	☐	✓

Rule Name: MAXIFACECOI Collect ifacecontrol records

Userdefined? ☐

Enabled? ✓

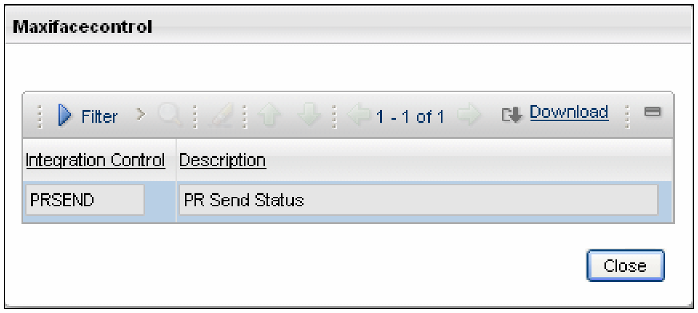
Primary Object: MAXIFACECOI

SQL Where:

```
ifacecontrol = 'PRSEND'
```

Figure 11-9 MAXIFACECONTROL Lookup rule

3. Validate that the appropriate object is selected by the validation process by clicking **Show Lookup Records**. The PRSEND integration control is displayed, as shown in Figure 11-10.


Maxifacecontrol

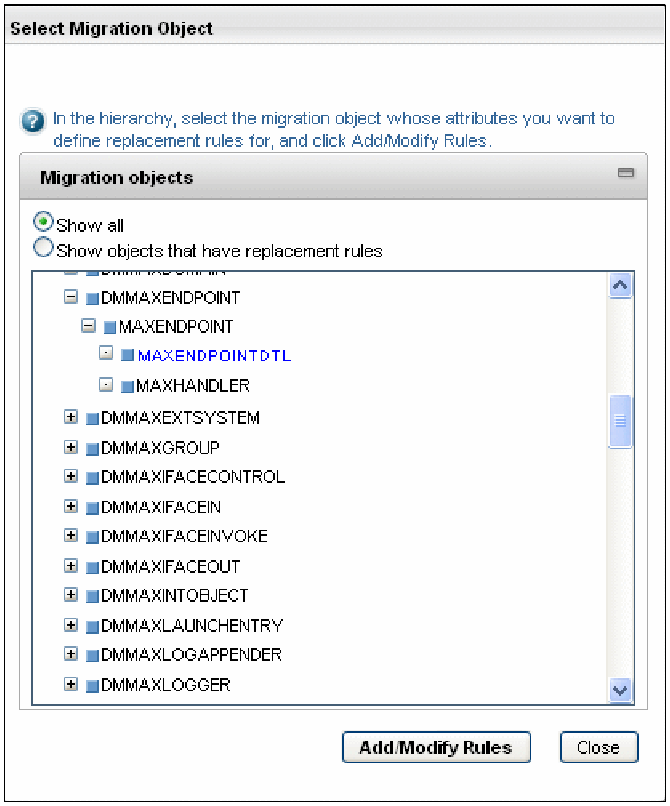
Integration Control	Description
PRSEND	PR Send Status

Figure 11-10 Show Lookup Records

Replacement rules

The endpoint that is created in source environment was configured to call the PR web service on the external system's `extdev` server. When this endpoint is migrated to the target environment, the endpoint should be configured to call the PR web service on the `extqa` server. To keep the configuration in the source environment intact and to modify the configuration for the target environment, we create a replacement rule in the target product environment's Migration Manager application.

Click **Go To** → **Migration** → **Migration Manager**. Select the **Add/Modify Replacement Rules** action. Browse the list of migration objects down to DMMAXENDPOINT → MAXENDPOINT → MAXENDPOINTDTL and select Add/Modify Rules, as shown in Figure 11-11.


Select Migration Object

In the hierarchy, select the migration object whose attributes you want to define replacement rules for, and click Add/Modify Rules.

Migration objects

[✓ Show all]
Show objects that have replacement rules

- DMMAXENDPOINT
  - MAXENDPOINT
    - MAXENDPOINTDTL
    - MAXHANDLER
- DMMAXEXTSYSTEM
- DMMAXGROUP
- DMMAXIFACECONTROL
- DMMAXIFACEIN
- DMMAXIFACEINVOKE
- DMMAXIFACEOUT
- DMMAXINTOBJECT
- DMMAXLAUNCHENTRY
- DMMAXLOGAPPENDER
- DMMAXLOGGER

Figure 11-11 Add/Modify Replacement Rules

Create the rule as shown in Table 11-8.

Table 11-8 Replacement rule MYPRWEBSERVICEENDPT

Field	Value
Rule	MYPRWEBSERVICEENDPT
Description	My PR Web Service End point
Sequence	1
Enabled	Y
Apply on Primary Object Insert?	Y
Apply on Primary Object Update?	Y
Apply on Primary Object Delete?	Y

Table 11-9 shows the replacement rule MYPRWEBSERVICEENDPT attributes.

Table 11-9 Replacement rule MYPRWEBSERVICEENDPT attributes

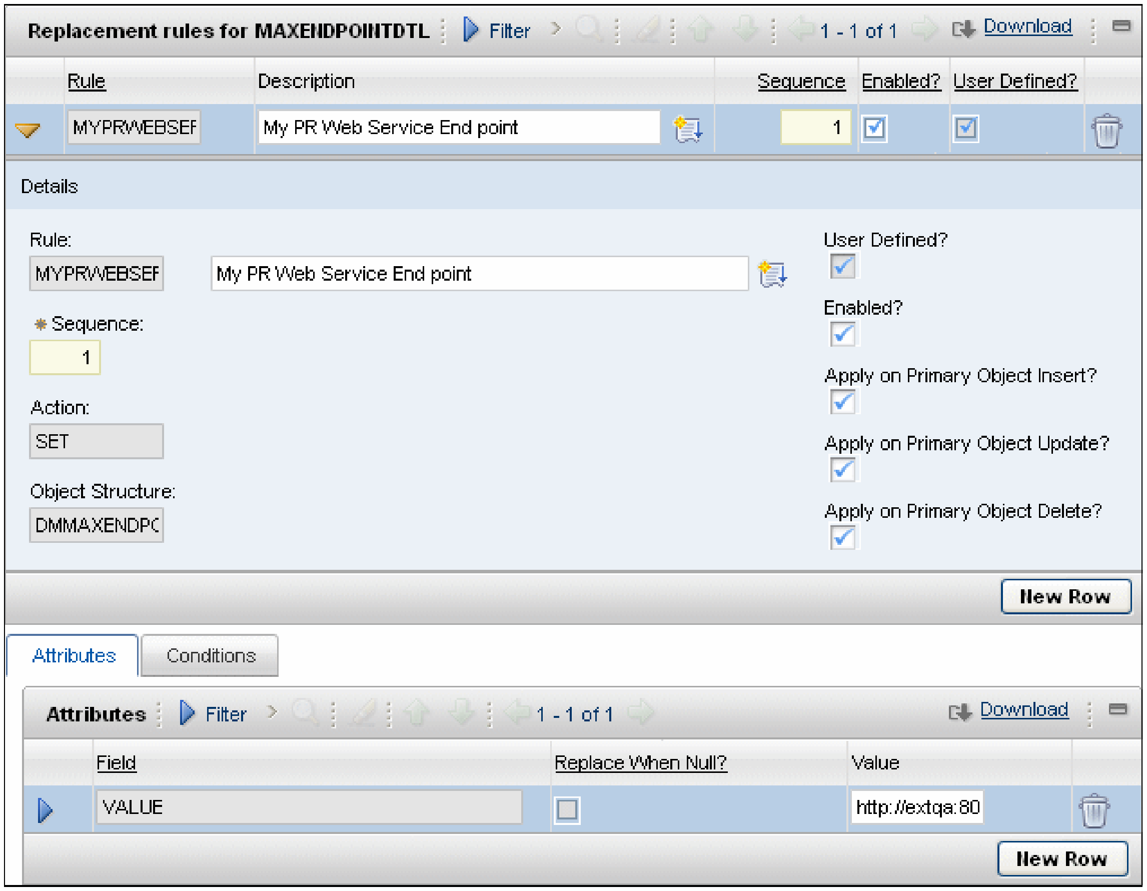
Field	Value
Field	VALUE
Value	http://extqa:8088/ExtPRWS
Replace When Null?	N

Table 11-10 shows the replacement rule MYPRWEBSERVICEENDPT conditions.

Table 11-10 Replacement rule MYPRWEBSERVICEENDPT conditions

Field	Value
Field	PROPERTY
Evaluation Type	EQUALS
Value	ENDPOINTURL

The replacement rule is shown in Figure 11-12.


Replacement rules for MAXENDPOINTDTL

Rule	Description	Sequence	Enabled?	User Defined?
MYPRWEBSEF	My PR Web Service End point	1	✓	✓

Details

Rule: MYPRWEBSEF My PR Web Service End point

Sequence: 1

Action: SET

Object Structure: DMMAXENDPC

User Defined? ✓

Enabled? ✓

Apply on Primary Object Insert? ✓

Apply on Primary Object Update? ✓

Apply on Primary Object Delete? ✓

Attributes

Field	Replace When Null?	Value
VALUE	☐	http://extqa:80

Figure 11-12   MYPRWEBSERVICEENDPT replacement rule

11.3.2 Creating the migration collection

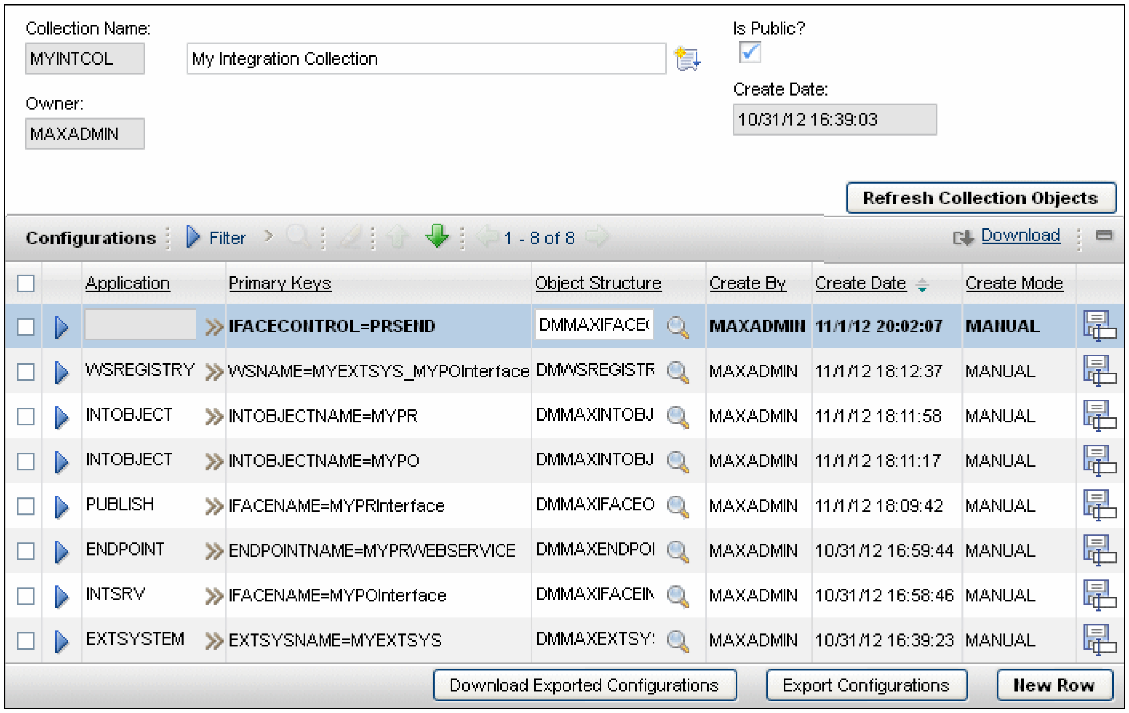
To create the migration collection, click **Go To** → **System Configuration** → **Migration** → **Migration Collections**. Create a migration collection and name it MYINTCOL.

The collection can be populated by completing the following steps for each integration component:

1. Create a row under the Configurations section.
2. Click **detail menu**, which is next to the Application field.
3. Find the wanted application in the list and click the **Go To** Application arrow.
4. Within the application, search for the wanted records. In this scenario, all records can be found by filtering on the key value MY.
5. Select the record and then select **Return With Value**.

After each record is selected, run validation. Validation automatically adds the PRSEND integration control to the collection, as specified in the lookup rule that was defined in the , “Lookup rules” on page 287.

The collection appears, as shown in Figure 11-13.


Collection Name: MYINTCOL — My Integration Collection

Owner: MAXADMIN

Is Public? ✓

Create Date: 10/31/12 16:39:03

Configurations — 1 - 8 of 8

Application	Primary Keys	Object Structure	Create By	Create Date	Create Mode
	IFACECONTROL=PRSEND	DMMAXIFACE(	MAXADMIN	11/1/12 20:02:07	MANUAL
WSREGISTRY	WSNAME=MYEXTSYS_MYPOInterface	DMWSREGISTR	MAXADMIN	11/1/12 18:12:37	MANUAL
INTOBJECT	INTOBJECTNAME=MYPR	DMMAXINTOBJ	MAXADMIN	11/1/12 18:11:58	MANUAL
INTOBJECT	INTOBJECTNAME=MYPO	DMMAXINTOBJ	MAXADMIN	11/1/12 18:11:17	MANUAL
PUBLISH	IFACENAME=MYPRInterface	DMMAXIFACEO	MAXADMIN	11/1/12 18:09:42	MANUAL
ENDPOINT	ENDPOINTNAME=MYPRWEBSERVICE	DMMAXENDPOI	MAXADMIN	10/31/12 16:59:44	MANUAL
INTSRV	IFACENAME=MYPOInterface	DMMAXIFACEIN	MAXADMIN	10/31/12 16:58:46	MANUAL
EXTSYSTEM	EXTSYSNAME=MYEXTSYS	DMMAXEXTSY:	MAXADMIN	10/31/12 16:39:23	MANUAL

Figure 11-13 MYINTCOL migration collection

The migration manager package is created by using the Create Package Definition action in the Migration Collections application. A link to the migration manager package is available for quick access from the Package Definitions tab.

In the Migration Manager application, the package is created in approved status. The migration groups DATADICTIONARY and INTEGRATION are automatically added to the migration package definition. Set the distribution target and activate the package. Create and distribute the package.

11.4 Deployment

The integration configurations can be deployed to the target environment with a single migration package.

Upload and deploy the migration package in the target environment. It is not necessary to enable Admin mode, redeploy enterprise archive (EAR) files, or restart the server.

The replacement rules take effect upon deployment; therefore, the endpoint has its ENDPOINTURL property replaced when the package is deployed. Because the external system was packaged in the source environment on an enabled state, the external system is created as enabled in the target environment. The outbound purchase requisition integration is configured and ready to produce outbound transactions in the target environment immediately after deployment.

The web service MYEXTSYS_MYPOInterface is created and automatically deployed within the Web Service Library. This deployment is possible because the web service is deployed to the product web service container instead of the application server web service container. The purchase order web service can accept transactions immediately after package deployment.

With the configurations performed for the integration and the migration in this chapter, the integration can be migrated to the target environment ready to send and accept transactions without the need to perform any manual steps after deployment or interrupt user activity.

Important: If you create a publish channel or enterprise service with processing rules on multiple objects, you might encounter the following error on package deployment:

```
BMXAA0048W - The sequence 1 already exists. Specify a unique
sequence.
```

In this case, the package is put into a status of DEPLOY_ERROR. To resolve the error, you can edit the package data in the Migration Manager application. The Deployment Data Errors tab is accessible under the Package tab in Migration Manager. It displays the error message for the migration object and the XML error data. For example, if the error occurs for a publish channel, the migration object is DMMAXIFACEOUT and the MAXIFACEOUT XML is displayed. Click **Edit Error Data**. Search for all instances of the PROCSEQUENCE element. You might receive this error because the content of PROCSEQUENCE is unique for each object but it is not unique for the entire publish channel. Renumber the processing rules' sequences so that they are unique for the entire publish channel by editing the contents of the PROCSEQUENCE elements. When you are finished editing the XML Error Data, click **Continue Deployment** for the migration package. The package deploys successfully.

11.5 Deployment considerations

In this use case, we demonstrated how to migrate an integration by using a simple pick and choose approach to build the migration collection.

11.5.1 Validation properties

The collection validation properties were configured to minimize the validation run time. This configuration was a beneficial to this particular use case because we were confident in the configuration objects that needed to be migrated. The validation process can be fine-tuned according to the developer's preferences by using these system properties.

11.5.2 Replacement rules

The replacement rule that was created in this scenario allowed for the purchase requisition's web service endpoint URL to point to the external system's development environment on our source environment. It also pointed to the external system's test environment on our target environment. This replacement rule eliminated the need to manually modify the endpoint's URL. If further development changes are required on the source environment, the rule is in place for migrating those changes to the target environment.

11.5.3 System availability

The DATADICTIONARY migration group is included in the migration manager package definition because we included two object structures in the collection. If migrating object structures, the inclusion of the DATADICTIONARY migration group in the package does not necessitate turning on Admin mode when the package is deployed.

Web services can be deployed to the product web service container or the application web service container. The product web service container was chosen in this case to avoid system downtime. To deploy web services to the product web service container, the system property `mxe.int.containerdeploy` must be set to `0`. This system property does not support live refresh; therefore, it requires a one-time restart of the application server. However, deploying web services to the application server web service requires rebuilding the EAR files and restarting the server every time a new web service is deployed.

In this use case, it is a prerequisite that the `mxe.int.containerdeploy` system property and the other relevant integration system properties and cron tasks are configured in the source environment and the target environment before the integration is developed in the source environment and the migration package is deployed in the target environment.

These factors were key to preserving system availability to the users during package deployment. Whether this package is deployed on a single server or a cluster environment, there is virtually no impact to the users during deployment.

12

Migrating service level agreements

Service level agreements (SLAs) as implemented in the Service Provider add on product present unique challenges during migrations. They include hidden dependencies that are often overlooked during the planning of a migration. Time is wasted trying to track down missing dependencies so migration packages can be rebuilt and redeployed. In this chapter, we describe how these dependencies can be found automatically so your migrations can succeed on the first attempt.

This chapter includes the following sections:

- Migrating service level agreements
- Deployment
- Deployment considerations

12.1 Migrating service level agreements

SLAs represent an agreement between a service provider and a service consumer. SLAs define commitments by the service provider to take certain actions within a certain amount of time. SLAs can be applied to many different object types in Maximo, and monitor that critical actions that are taken on those objects happen in a timely manner. SLAs use escalations, actions, and communication templates to take predefined actions and send notifications when a violation of the SLA commitments occurs.

12.1.1 Requirements

SLAs can have dependencies on the following types of objects:

- Escalations
- Actions
- Action groups
- Communication templates
- Roles
- Persons
- Person groups
- Organizations
- Sites
- Calendars

These objects must be deployed in the target environment before an SLA that depends on them can be deployed. Deploying the SLA first can fail when the SLA validation notices the references to missing objects.

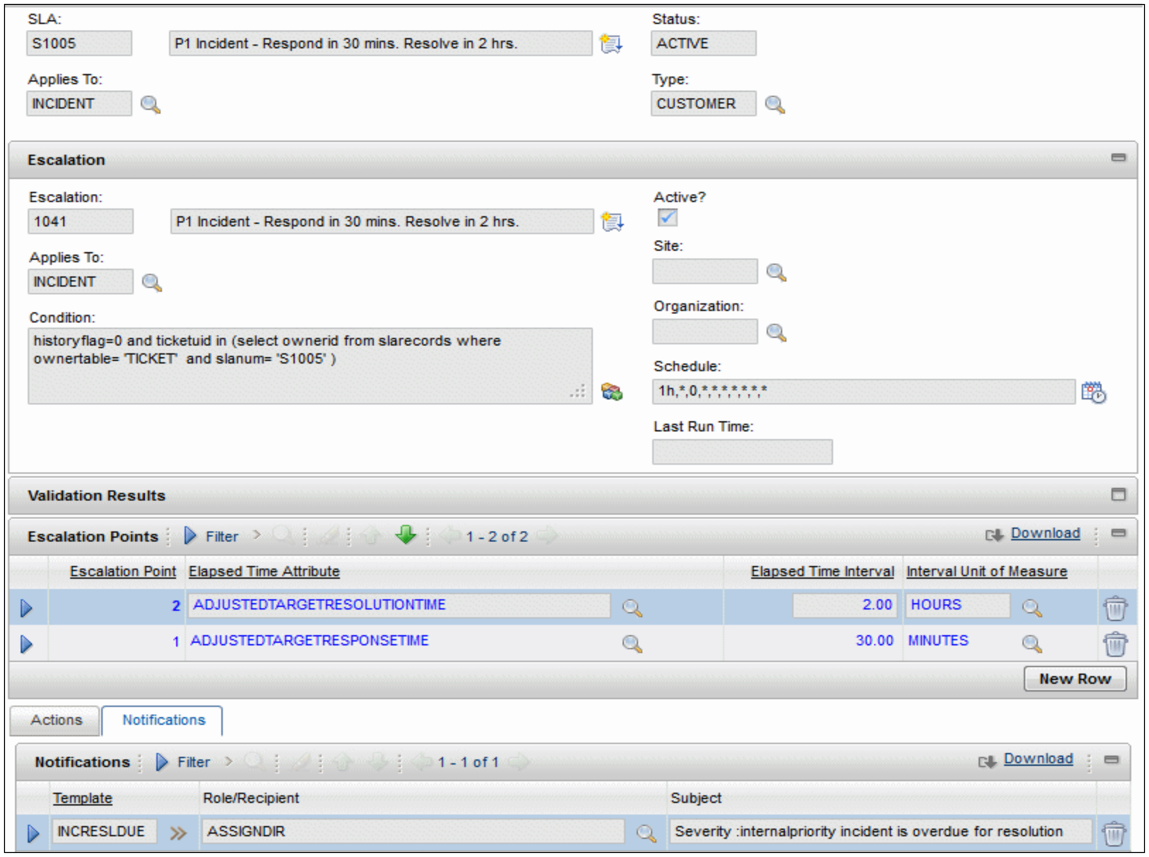
The dependency on escalations presents a unique migration challenge. An SLA's escalation is edited by using the SLAs application rather than the Escalations application. Configuration developers who are working on SLAs often overlook the fact that they created or modified escalations and tell the person who is tasked with performing the migrations only, “Migrate SLA 1204.”

The following sections show how migration collections and the collection validation process can be used to automatically find and migrate any escalations that are associated with SLAs that are migrated. The process also finds any actions, action groups, communication templates, and roles that are associated with the escalations.

12.1.2 Configuration applications

You can define, manage, and delete classifications in the Classifications application. You access the Classifications application by clicking **Go To** → **Service Level** → **Service Level Agreements**.

Figure 12-1 shows an example of an escalation in the SLA application. Users might forget that changes that are made here are not changes to the SLA.



SLA:	S1005 — P1 Incident - Respond in 30 mins. Resolve in 2 hrs.
Status:	ACTIVE
Applies To:	INCIDENT
Type:	CUSTOMER

Escalation

Escalation:	1041 — P1 Incident - Respond in 30 mins. Resolve in 2 hrs.
Active?	✓
Applies To:	INCIDENT
Site:	
Organization:	
Condition:	historyflag=0 and ticketuid in (select ownerid from slarecords where ownertable= 'TICKET' and slanum= 'S1005')
Schedule:	1h,*,0,*,*,*,*,*,*,*
Last Run Time:	

Validation Results

Escalation Points Filter 1 - 2 of 2 Download

Escalation Point	Elapsed Time Attribute	Elapsed Time Interval	Interval Unit of Measure
2	ADJUSTEDTARGETRESOLUTIONTIME	2.00	HOURS
1	ADJUSTEDTARGETRESPONSETIME	30.00	MINUTES

New Row

Actions | Notifications

Notifications Filter 1 - 1 of 1 Download

Template	Role/Recipient	Subject
INCRESLDUE	ASSIGNDIR	Severity :internalpriority incident is overdue for resolution

Figure 12-1 SLA application allows editing of related escalation

Before migrating SLAs, be sure to migrate organizations, persons, calendars, classifications, and service groups if you used these objects in SLA definitions.

12.1.3 Object structure

You can use the predefined DMPLUSPSLA object structure, as shown in Figure 12-2.

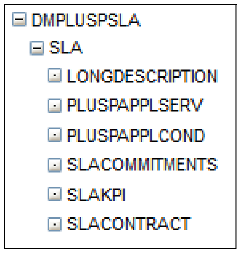


Figure 12-2 DMPLUSPSLA object structure

12.1.4 Migration group

A migration collection automatically selects migration groups for use when a migration package definition is created that is based on the collection. It prefers migration groups that are marked as internal and then prefers the group with the lowest migration group order value.

There are no internal migration groups that contain the DMPLUSPSLA object structure. The lowest-numbered group that contains this object structure is PLUSPSERVICEPROVIDER. There is no need to create your own migration group unless you created your own SLA object structure.

12.1.5 Migration collection

This section describes how to create the migration collection that is used to migrate SLAs and the objects on which they depend.

Complete the following steps to create a migration collection:

1. Click **Go To** → **System Configuration** → **Migration** → **Migration Collections**.
2. Create a migration collection.
3. Give it a name, such as SLA.
4. Select the **Is Public** option if you want other users to add objects to this collection.
5. Add SLAs to the collection.

6. Select **Validate Migration Collection** from the Select Action menu, or click the icon on the toolbar. Start the validation and wait for it to complete. Refresh the validation results.

   No related configurations are displayed.

No related configurations are displayed because there are no internal migration groups that contain migration object structures for SLAs. The default validation algorithm finds only related configurations if the collection object's object structure is used in an internal migration group, and the related configuration's object structure is used by a migration group with a higher migration group order.

It is necessary to explicitly define a related data rule that defines how the validation algorithm can find escalations that are related to SLAs in the collection. To define a related data rule, select **Define Related Data and Lookup Rules** from the Select Action menu of the Migration Collections application. Then, select the **Related Data Rules** tab.

A number of related data rules are predefined, but you must add a new rule to define how to find an SLA's escalation. Click **New** to add a new rule. Figure 12-3 shows the fields for a related data rule.

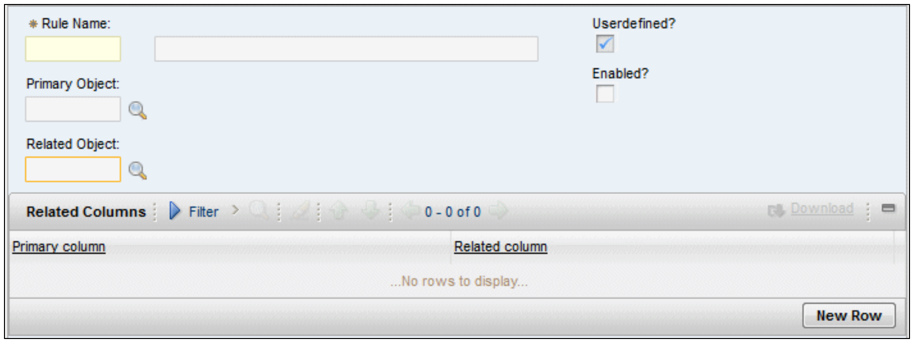


Figure 12-3 Related Data Rule fields

The Primary Object field contains the name of an object type. The rule looks for objects of this type that are members of the migration collection. The Related Object field contains the name of an object type that can be related to the primary object. Entries in the Related Columns table define the columns in the primary object that match the columns in the related object. These entries are used to construct queries that find related objects for the primary objects that are in the collection. The resulting related object sets are added to the collection's Related Configurations list.

Click the **Select Value** icon that is next to the Related Object field. A list of all object types that can legally be used as related objects is displayed. This list is short in a default installation. The list does not include escalations because the definition of the domain on the related object attribute. The default domain on this attribute excludes all object types that are the main types for an application. Because of this configuration, escalations are not available for use as the related object in a rule.

Changing the related objects domain

Before you can define a rule that finds escalations that are associated with SLAs, you must first change the domain for the related object attribute on the rule object. You cannot make this change by editing the domain in the Domains application. The domain is marked as internal and the Domains application does not edit internal domains. You must create a domain that allows the selection of object types that are main types for applications.

The Database Configuration application can be used to change an attribute's domain, but you cannot change an object type that is marked as internal. The related data rule object type is internal, so you must clear the internal setting on the DMCOLLRELRULE object type before you can change the domain on the related object attribute. You cannot make this change through the Maximo user interface. You must make this change directly to the Maximo database by using SQL statements.

Complete the following steps to create a domain for the related objects field:

1. Connect to the Maximo database by using the SQL client of your choice. Run the following two commands to clear the internal setting on the DMCOLLRELRULE object type:

   ```
   update maxobject set internal = 0 where objectname = 'DMCOLLRELRULE'
   update maxobjectcfg set internal = 0 where objectname =
   'DMCOLLRELRULE'
   ```

   Disconnect from the database.
2. Open the Maximo Database Configuration application.
3. Search for the DMCOLLRELRULE object type.
4. Go to the Attributes tab and find the RELATEDOBJECT attribute. Open the details pane for this attribute.

   The attribute's domain should be set to DMCOLLRELOBJECT.
5. Click the **Detail Menu** icon that is next to the attribute's Domain field and select Go To Domains.

   The Domains application opens with the DMCOLLRELOBJECT domain loaded.

6. Click the **Edit Details** icon that is to the left of the domain name.
7. Open the details pane for the domain.

   DMCOLLRELOBJECT is an internal domain. You cannot modify its definition. But, you can copy and paste the definitions of the domain's validation and list clauses into a temporary document and use these definitions to create a domain with modified versions of these clauses.
8. Copy the values of the Validation Where Clause and List Where Clause fields into a temporary text document.

   The validation clause should include the following value:

   ```
   objectname = :relatedobject and objectname in (select a.objectname
   from maxintobjdetail a, maxintobject b  where
   maxobject.objectname=a.objectname and a.intobjectname
   =b.intobjectname and a.parentobjid is null and (a.objectname not in
   (select value from alndomain where domainid='DMCOLLLOOKUP')) and
   a.objectname not in (select maintbname from maxapps where maintbname
   is not null ) and b.usewith in (select value from synonymdomain
   where domainid='INTUSEWITH' and maxvalue='MIGRATIONMGR'))
   ```

   The list clause should include the following value:

   ```
   exists (select a.objectname from maxintobjdetail a, maxintobject b
   where maxobject.objectname=a.objectname and a.intobjectname
   =b.intobjectname and a.parentobjid is null and (a.objectname not in
   (select value from alndomain where domainid='DMCOLLLOOKUP')) and
   a.objectname not in (select maintbname from maxapps where maintbname
   is not null ) and b.usewith in (select value from synonymdomain
   where domainid='INTUSEWITH' and maxvalue='MIGRATIONMGR'))
   ```

   The underlined portions of the clauses are what causes object types that are used as the main types of applications to be excluded from the related data rule's related object field. Remove the underlined portions from the clauses in your temporary text document.
9. Cancel the Table Domain window that shows the DMCOLLRELOBJECT domain.
10. Click **Add New Domain** at the lower right of the Domains application. Select the Add New TABLE Domain option.
11. Give the new domain a name, such as XDMCOLLRELOBJECT.
12. Click **New Row** in the Table Domain list.
13. Select MAXOBJECT as the object type.

14. Copy and paste the edited WHERE clauses from your temporary text document into the Validation Where Clause and List Where Clause fields. Ensure that you removed the underlined portions of the WHERE clauses.

    You are now done with the temporary text document.

15. Click **OK** to close the Table Domain window. Click **Save**.
16. Click **Return With Value** to return the new domain to the Database Configuration application.

    You should now be back in the Database Configuration application with your new domain entered in the Domain field of the RELATEDOBJECT attribute.

17. Click **Save** to save the change to the RELATEDOBJECT attribute's domain.
18. Click **View Record List** to return to the Database Configuration application's list view.
19. Use the Select Action menu to enable Admin mode, apply configuration changes, and disable Admin mode.

Creating a related data rule

You are now ready to create a related data rule that finds escalations that belong to SLAs.

1. Open the Migration Collections application. Search for the collection to which you added the SLAs to be migrated.
2. Select **Define Related Data** and **Lookup Rules** from the Select Action menu of the Migration Collections application. Then, select the **Related Data Rules** tab.
3. Click **New Row** to define a new rule.
4. Give the new rule a name, which indicates that it searches for escalations that belong to SLAs, such as SLAESC.
5. Select **SLA** as the primary object.
6. Select **ESCALATION** as the related object.
7. Click **New Row** from the Related Columns list.
8. Enter `ESCALATION` in the Primary Column and Related Column fields.

    The SLA object type features a column that is named ESCLATION. This column contains the name of an escalation if the SLA has an escalation. The escalation object type stores escalation names in a column that is also named ESCALATION.

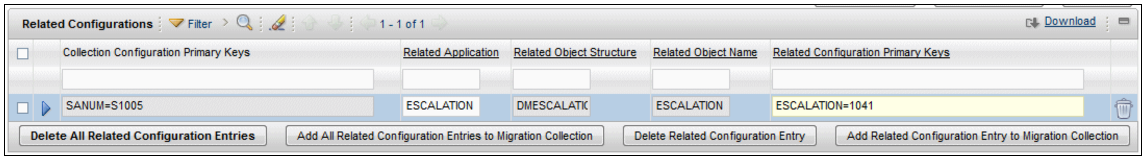
    When the collection validation algorithm evaluates this rule, it creates a query with the WHERE clause ESCALATION.ESCALATION = SLA.ESCALATION to find all escalations that are associated with each SLA in the collection.

9. Select **Enabled**.
10. Click **OK** to save the rule.

Rerun collection validation

Collection validation found no related configurations when it was run before the related data rule was created. Run a validation again. Remember to refresh the validation results after the validation completes.

You should now see escalations in the related configurations list if any of the SLAs in your collection include escalations. Figure 12-4 shows the validation results for a collection that contains a single SLA named S1005 where this SLA has an escalation named 1041.


Related Configurations
Filter
1 - 1 of 1
Download
Delete All Related Configuration Entries
Add All Related Configuration Entries to Migration Collection
Delete Related Configuration Entry
Add Related Configuration Entry to Migration Collection

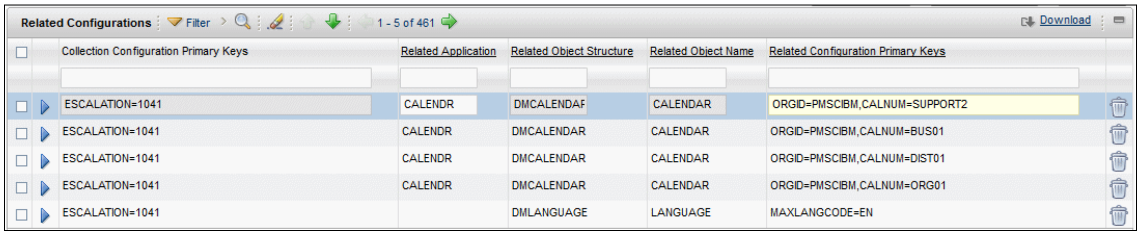
Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
SANUM=S1005	ESCALATION	DMESCALATIC	ESCALATION	ESCALATION=1041

Figure 12-4 Collection validation finds an escalation

You want any escalations that were found by the validation process to be included in the migration, so click **Add All Related Configuration Entries to Migration Collection**. These entries disappear from the related configuration list and appear in the Configurations list on the Migration Collections tab.

The collection still does not contain any objects on which the newly added escalations might depend. You must run the validation process again so the validation algorithm can look for configurations that are related to the escalations. This time, there is no need to add any new related data rules. Escalations do have object structures that are referenced by internal migration groups. This configuration allows the default validation processing to find items that are related to escalations.

Figure 12-5 shows the related configurations that are found after the collection that contains SLA S1005 and escalation 1041 is validated.



Related Configurations Filter 1 - 5 of 461 Download

Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
ESCALATION=1041	CALENDR	DMCALENDAF	CALENDAR	ORGID=PMSCIBM,CALNUM=SUPPORT2
ESCALATION=1041	CALENDR	DMCALENDAR	CALENDAR	ORGID=PMSCIBM,CALNUM=BUS01
ESCALATION=1041	CALENDR	DMCALENDAR	CALENDAR	ORGID=PMSCIBM,CALNUM=DIST01
ESCALATION=1041	CALENDR	DMCALENDAR	CALENDAR	ORGID=PMSCIBM,CALNUM=ORG01
ESCALATION=1041		DMLANGUAGE	LANGUAGE	MAXLANGCODE=EN

Figure 12-5 Collection validation finds items that are related to an escalation

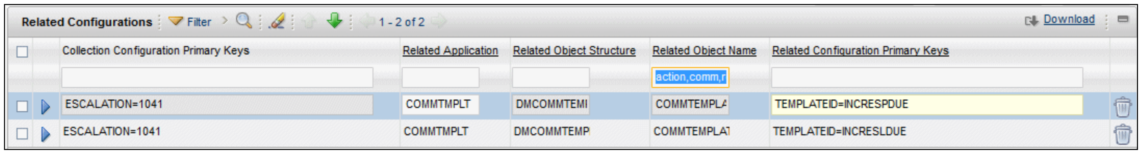
The validation process identifies many related objects that you are probably not interested in migrating. It identifies organizations, sets, calendars, object types, relationships, and many other objects that are related to the escalations, but are unlikely to be created or modified as a part of defining the escalation. You can filter this list to show only the objects that are likely to break the migration of the escalation if they are left out of the migration.

Escalations often reference actions, action groups, and communication templates that were created only for the escalation. These objects must be included in the collection if they were not migrated before. Leaving them out causes the deployment of the escalation to fail when it is discovered that the escalation references objects do not exist in the deployment environment.

Communication templates often refer to roles that might be created or modified when the template was created. These roles must be included in the collection or the deployment of the communication templates might fail.

You can filter the Related Configurations list to show only actions, action groups, communication templates, and roles by entering the `action,comm,role` filter in the Related Object Name field of the list. Each comma-separated portion of the filter string is used to match the beginning of object names. So `action` matches ACTION and ACTIONGROUP, and `comm` matches COMMTEMPLATE.

Figure 12-6 shows the result of applying this filter to the validation results of the example collection. Escalation 1041 does not feature any actions or action groups, but it does include two communication templates.



Related Configurations | Filter | 1 - 2 of 2 | Download

Collection Configuration Primary Keys	Related Application	Related Object Structure	Related Object Name	Related Configuration Primary Keys
			action,comm,r	
ESCALATION=1041	COMMTMPLT	DMCOMMTEMI	COMMTEMPLA	TEMPLATEID=INCRESPDUE
ESCALATION=1041	COMMTMPLT	DMCOMMTEMP	COMMTEMPLAT	TEMPLATEID=INCRESLDUE

Figure 12-6 Validation results filtered to show communication templates

Select any actions, action groups, or communication templates that are shown in your filtered results and click **Add Related Configuration Entry to Migration Collection**.

Run the validation process again, which is necessary to find any actions that are referenced by action groups that were found in the prior validation, and to find any roles that are referenced by communication templates. The filter that you applied to the Related Configurations list is still in place, so there is no need to filter again.

In the sample collection, a single role is used by both communication templates. This role appeared in the new validation results and was added to the collection. Running validation again now finds no new objects that match the filter.

You are now ready to save the collection, generate a migration package definition, and create a migration package.

12.2 Deployment

After the package is created, you must distribute the package to distribute it to the target environment. You can distribute the package via file-type distribution or database-type distribution. For more information, see Chapter 15, “Common topics” on page 351.

SLAs can reference many object types that were not described in the sample migration scenario. These types include, but are not limited to, the following types:

- Organizations
- Calendars
- Shifts
- Service groups
- Classifications
- Locations

Ensure that any such objects that are referenced by the SLAs you are migrating exist in the deployment environment. These objects often are migrated in their own collections rather than in the SLA collection.

12.3 Deployment considerations

There are special considerations to deploying escalations. These considerations also apply to SLAs that include escalations.

Escalations include associated cron task instances that must exist in the deployment environment for the escalations to work. The easiest way to accomplish this configuration is to disable the escalation before a migration package that includes it is created. You can then re-enable the escalation. When the escalation is deployed, it is in the same disabled state it was in when the deployed package was created. You can manually enable the escalation after it is deployed, which automatically creates the cron task instance it needs.

There is no need to deal with the escalation states directly when you are migrating SLAs. Instead, you can change the state of the SLAs you are migrating to INACTIVE before the migration package that contains them is created. This configuration automatically disables any escalations that are associated with the SLAs. After the SLAs are deployed, you can change their states to ACTIVE. This change in state activates any escalations that are associated with the SLAs and creates the cron task instances that are needed by the escalations.

13

Migrating service offering content

This chapter provides you with an overview of the process steps, issues, and concerns that we experienced when the IBM Service Offering content in the IBM SmartCloud Control Desk product was migrated.

Because the service offerings are technically content-type data, often you would move this data from one environment to another environment by using the Integration Framework. However, IBM provides specialized knowledge to the Integrated Service Management (ISM) marketplace by embedding the tool-specialized object structures to migrate the IBM-delivered content.

This chapter presents scenarios in which the source and target contain the IBM content. If your target environment does not contain the same product installations as your source environment regarding the content packages, see Chapter 16, “Troubleshooting” on page 419 for more information about how to manage these situations.

This chapter includes the following sections:

- Service offering scenario
- Requirements
- Solution A
- Solution B
- Deployment

13.1 Service offering scenario

The Service Offering content that IBM delivers with the SmartCloud Control Desk uses many objects in the database. The migration of these object structures requires attention to which service offerings you must migrate. The various migration groups that make up the predefined migration package template include a many various configurations and content-type object structures. Managing this migration requires you to enter the correct data and to include (or exclude) it by managing the WHERE clauses in the package definition.

13.2 Requirements

Migrate the other and modified service offerings in a SmartCloud Control Desk implementation by using the sample migration packages that are provided as templates for the various service offerings that are provided or developed.

13.3 Solution A

You can access the template migration package by browsing to the Migration Package application by clicking **Go To** → **System Configuration** → **Migration** → **Migration Manager**, as shown in Figure 13-1.

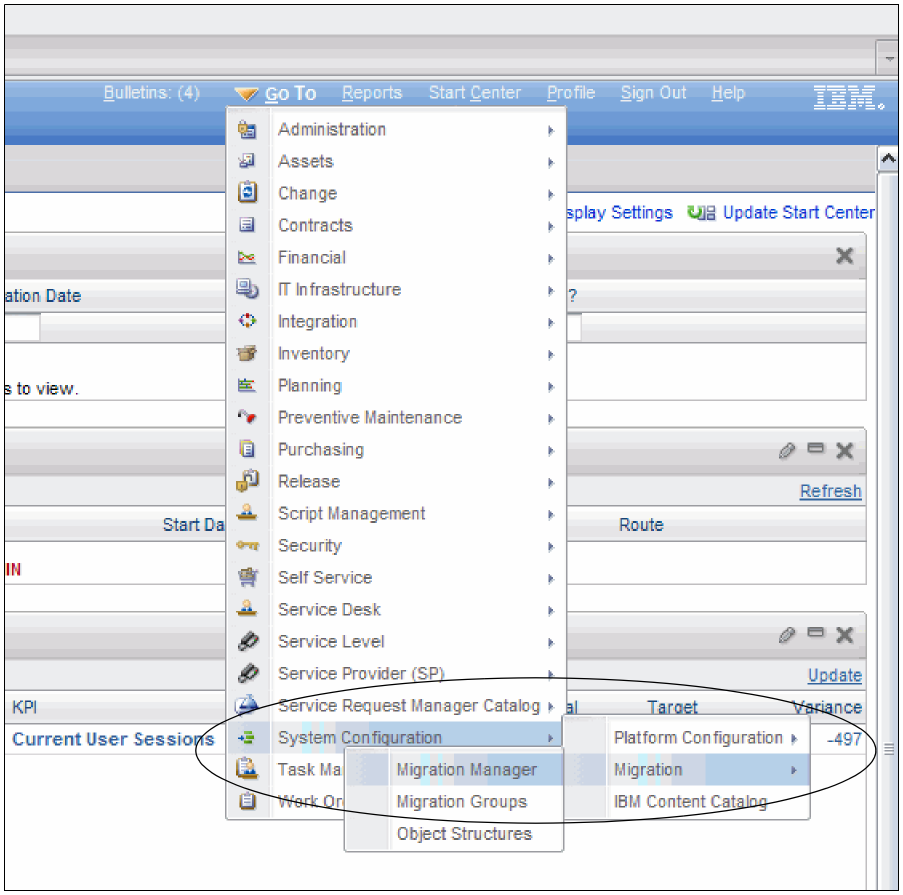
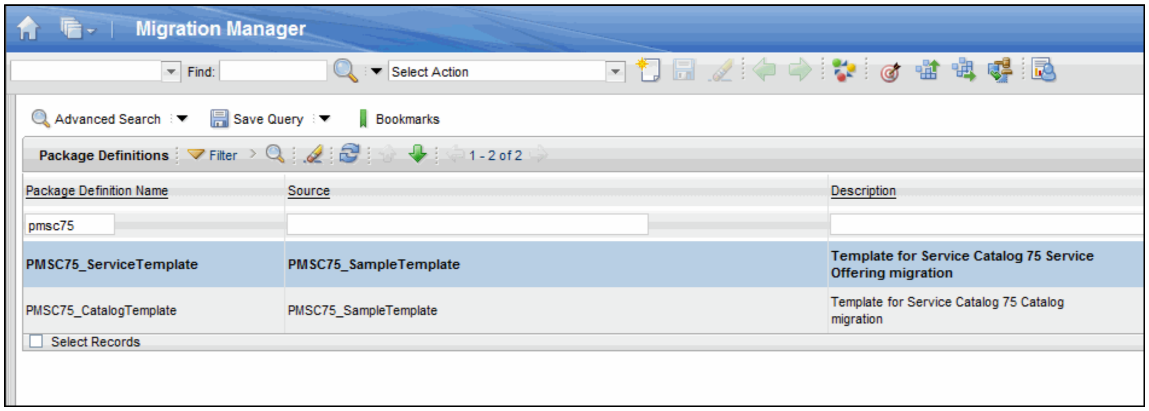


Figure 13-1 *Browsing to the Migration Manager application*

For this scenario, we need the PMSC75_ServiceTemplate package. Because the name of this package can differ from your exact environment, you can search for it, as shown in Figure 13-2.

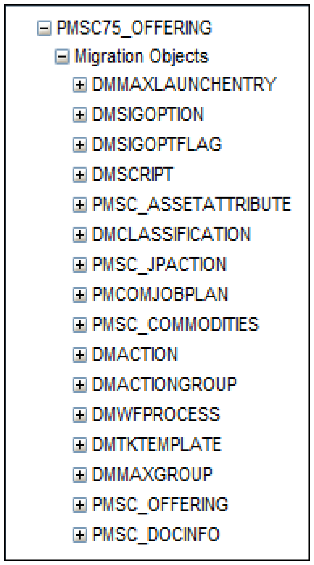


Package Definition Name	Source	Description
pmsc75		
PMSC75_ServiceTemplate	PMSC75_SampleTemplate	Template for Service Catalog 75 Service Offering migration
PMSC75_CatalogTemplate	PMSC75_SampleTemplate	Template for Service Catalog 75 Catalog migration

Figure 13-2 Service Template Offering

Important: There are two older PMSC72 Service Offering and Service Catalog Packages, which must not be used in version 7.5 environments.

When you examine the Migration Groups structure, you can see that there are many things to consider about this scenario, as shown in Figure 13-3.



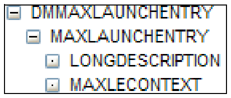
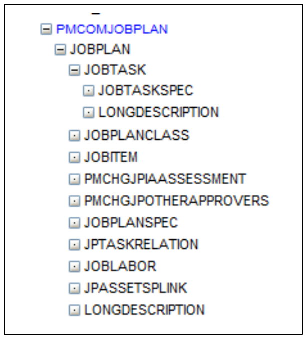
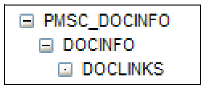
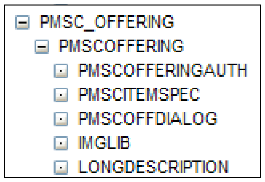
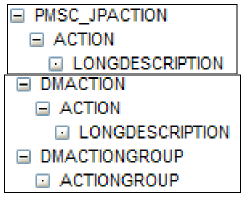
```
⊟ PMSC75_OFFERING
  ⊟ Migration Objects
    ⊞ DMMAXLAUNCHENTRY
    ⊞ DMSIGOPTION
    ⊞ DMSIGOPTFLAG
    ⊞ DMSCRIPT
    ⊞ PMSC_ASSETATTRIBUTE
    ⊞ DMCLASSIFICATION
    ⊞ PMSC_JPACTION
    ⊞ PMCOMJOBPLAN
    ⊞ PMSC_COMMODITIES
    ⊞ DMACTION
    ⊞ DMACTIONGROUP
    ⊞ DMWFPROCESS
    ⊞ DMTKTEMPLATE
    ⊞ DMMAXGROUP
    ⊞ PMSC_OFFERING
    ⊞ PMSC_DOCINFO
```

Figure 13-3 Migration group for service offering template

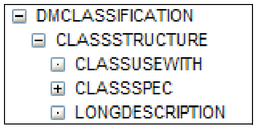
13.3.1 Configuration applications

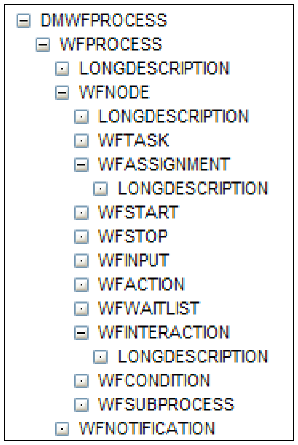
To understand the Migration Group, Table 13-1 shows the various applications that make up a service offering.

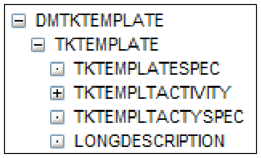
Table 13-1 Configuration applications

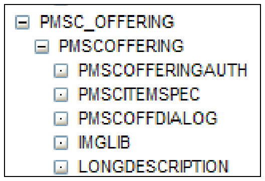
Application name	Object structure
Launch in Context	 DMMAXLAUNCHENTRY MAXLAUNCHENTRY LONGDESCRIPTION MAXLECONTEXT
Job Plans	 PMCOMJOBPLAN JOBPLAN JOBTASK JOBTASKSPEC LONGDESCRIPTION JOBPLANCLASS JOBITEM PMCHGJPIAASSESSMENT PMCHGJPOTHERAPPROVERS JOBPLANSPEC JPTASKRELATION JOBLABOR JPASSETSPLINK LONGDESCRIPTION
View Documents	 PMSC_DOCINFO DOCINFO DOCLINKS
Offerings	 PMSC_OFFERING PMSCOFFERING PMSCOFFERINGAUTH PMSCITEMSPEC PMSCOFFDIALOG IMGLIB LONGDESCRIPTION
Actions Note: There are two Object Structures for this application.	 PMSC_JPACTION ACTION LONGDESCRIPTION DMACTION ACTION LONGDESCRIPTION DMACTIONGROUP ACTIONGROUP

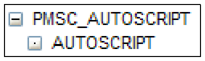
Application name	Object structure

Classifications
Classifications (SP)
	
	⊟ DMCLASSIFICATION
	  ⊟ CLASSSTRUCTURE
	    ⊡ CLASSUSEWITH
	    ⊞ CLASSSPEC
	    ⊡ LONGDESCRIPTION

Workflow Designer	
	⊟ DMWFPROCESS
	  ⊟ WFPROCESS
	    ⊡ LONGDESCRIPTION
	    ⊟ WFNODE
	      ⊡ LONGDESCRIPTION
	      ⊡ WFTASK
	      ⊟ WFASSIGNMENT
	        ⊡ LONGDESCRIPTION
	      ⊡ WFSTART
	      ⊡ WFSTOP
	      ⊡ WFINPUT
	      ⊡ WFACTION
	      ⊡ WFWAITLIST
	      ⊟ WFINTERACTION
	        ⊡ LONGDESCRIPTION
	      ⊡ WFCONDITION
	      ⊡ WFSUBPROCESS
	    ⊡ WFNOTIFICATION

Ticket Templates	
	⊟ DMTKTEMPLATE
	  ⊟ TKTEMPLATE
	    ⊡ TKTEMPLATESPEC
	    ⊞ TKTEMPLTACTIVITY
	    ⊡ TKTEMPLTACTYSPEC
	    ⊡ LONGDESCRIPTION

Service Groups	
	⊟ PMSC_OFFERING
	  ⊟ PMSCOFFERING
	    ⊡ PMSCOFFERINGAUTH
	    ⊡ PMSCITEMSPEC
	    ⊡ PMSCOFFDIALOG
	    ⊡ IMGLIB
	    ⊡ LONGDESCRIPTION

Automation Scripts	 ⊟ PMSC_AUTOSCRIPT ⊡ AUTOSCRIPT

You can use the following navigation links to access these applications:

- Launch In Context: Select **Go To** → **System Configuration** → **Platform Configuration** → **Launch in Context**.
- Job Plans: Select **Go To** → **Planning** → **Job Plans**.
- View Documents: Select **Go To** → **Administration** → **View Documents**.
- Offerings: Select **Go To** → **Service Request Manager Catalog** → **Offerings**.
- Actions: Select **Go To** → **System Configuration** → **Platform Configuration** → **Actions**.
- Classifications or Classifications (SP): Select **Go To** → **Administration** → **Classifications**.
- Workflow Designer: Select **Go To** → **System Configuration** → **Platform Configuration** → **Workflow Designer**.
- Ticket Templates: Select **Go To** → **Service Desk** → **Ticket Templates**.
- Service Groups: Select **Go To** → **Service Request Manager Catalog** → **Service Groups**.
- Automation Scripts: Select **Go To** → **Script Management** → **Automation Scripts**.

13.3.2 Object structures

This section describes only those object structures that are unique to the SmartCloud Control Desk (that are not distributed with Tivoli's process automation engine). These object structures are unique to the following migrating services that offer content:

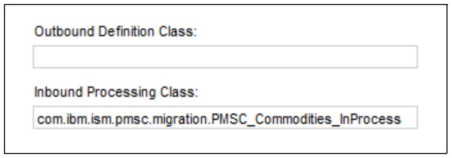
- PMSC_ASSETATTRIBUTE
- PMSC_COMMODITIES
- PMCOMJOBPLAN
- PMSC_DOCINFO
- PMSC_JPACTION
- PMSC_OFFERING

PMSC_ASSETATTRIBUTE

The PMSC_ASSETATTRIBUTE object structure brings in only one object - the ASSEATTRIBUTE object. This object is needed for the correct use of attributes that are associated with the job plans.

PMSC_COMMODITIES

The PMSC_COMMODITIES object structure allows for the data validation of items to the offering type within the correct itemset. The processing class is unique to the service offering content. Figure 13-4 shows the inbound processing class.



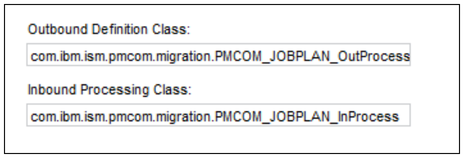
Outbound Definition Class:

Inbound Processing Class: com.ibm.ism.pmsc.migration.PMSC_Commodities_InProcess

Figure 13-4 PMSC_COMMODITIES object structure

PMCOMJOBPLAN

This object structure has classes for inbound and outbound processing that are unique to the service offering content. These classes are not used for other object structures or purposes. Figure 13-5 shows the identified classes.

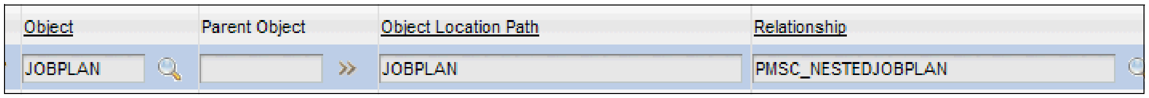


Outbound Definition Class: com.ibm.ism.pmcom.migration.PMCOM_JOBPLAN_OutProcess

Inbound Processing Class: com.ibm.ism.pmcom.migration.PMCOM_JOBPLAN_InProcess

Figure 13-5 Custom Tivoli Service Request Manager classes for Job Plan migration

IBM also delivers a new relationship (as shown in Figure 13-6) for the SmartCloud Control Desk product offering with which you can migrate the service offerings correctly.

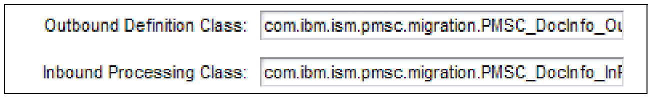


Object	Parent Object	Object Location Path	Relationship
JOBPLAN		JOBPLAN	PMSC_NESTEDJOBPLAN

Figure 13-6 PMSC nested job plan relationship

PMSC_DOCINFO

The SmartCloud Control Desk Service Offering content provides you with the Outbound Definition and Inbound Processing classes for the PMSC_DOCINFO object structure, as shown in Figure 13-7.



Outbound Definition Class: com.ibm.ism.pmsc.migration.PMSC_DocInfo_Ou

Inbound Processing Class: com.ibm.ism.pmsc.migration.PMSC_DocInfo_InP

Figure 13-7 PMSC_DOCINFO specialized classes

These classes enable the product to properly process content-type data with the migration of configuration data.

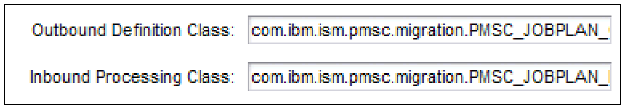
Important: This object structure is tightly coupled with the PMSCOFFERING object. If you try to use it for docinfo objects that are not owned by PMSCOFFERING, it fails in the creation phase. The WHERE clause that is provided with the product is shown in the following example:

```
docinfoid in (select docinfoid from doclinks where
ownertable='PMSCOFFERING' and ownerid in (select itemid from
pmscoffering where itemnum='SERVICENAME'))
```

We encourage you to review the definition separately by using the Database Configuration application.

PMSC_OFFERING

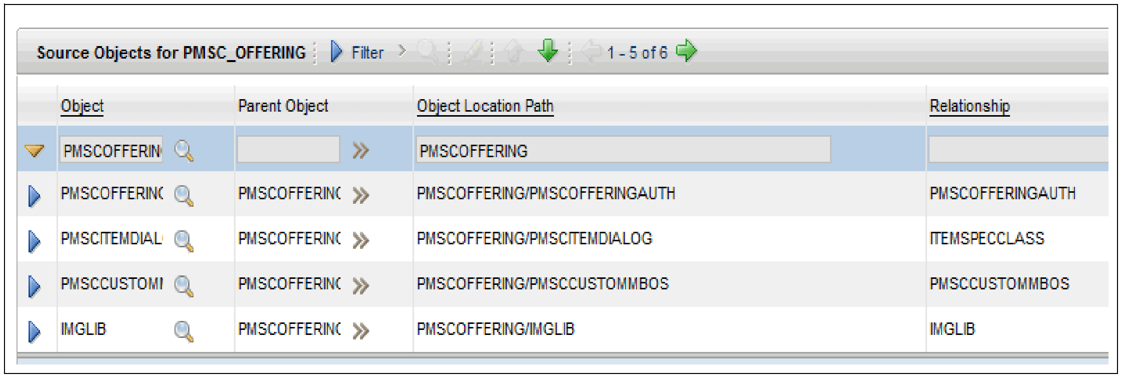
The PMSC_Offering object structure requires that you understand that other classes and relationships are used separately from the standard product, as shown in Figure 13-8.



Outbound Definition Class:	com.ibm.ism.pmsc.migration.PMSC_JOBPLAN_
Inbound Processing Class:	com.ibm.ism.pmsc.migration.PMSC_JOBPLAN_

Figure 13-8 PMSC_OFFERING specialized classes

The relationships for this object structure are shown in Figure 13-9.



Source Objects for PMSC_OFFERING   Filter   1 - 5 of 6

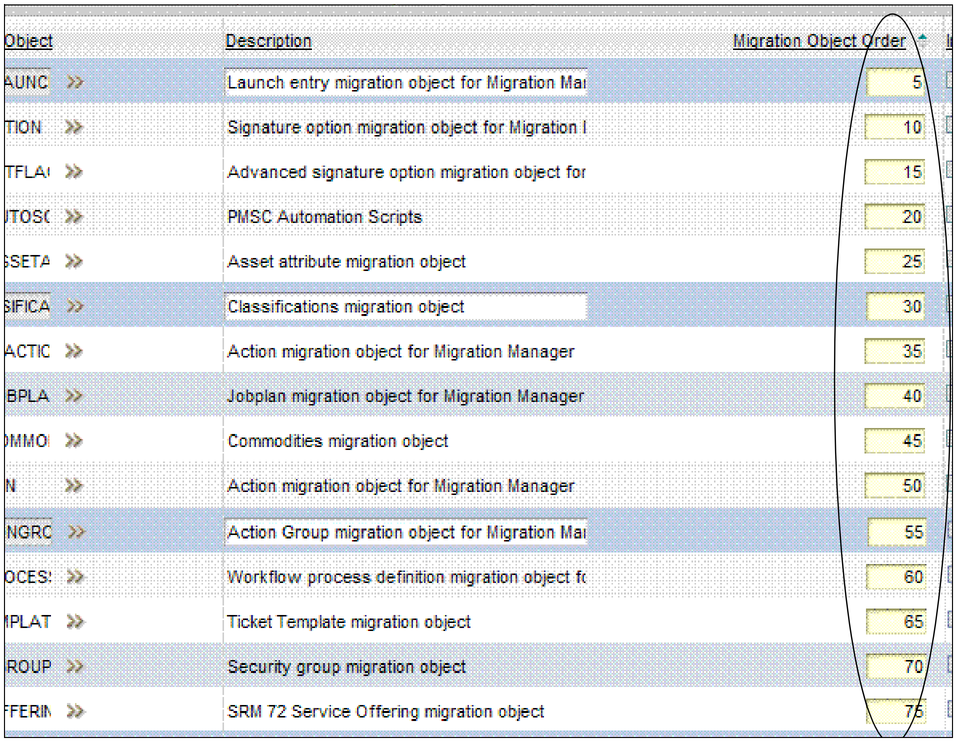
Object	Parent Object	Object Location Path	Relationship
PMSCOFFERIN		PMSCOFFERING	
PMSCOFFERINC	PMSCOFFERINC	PMSCOFFERING/PMSCOFFERINGAUTH	PMSCOFFERINGAUTH
PMSCITEMDIAL	PMSCOFFERINC	PMSCOFFERING/PMSCITEMDIALOG	ITEMSPECCLASS
PMSCCUSTOMI	PMSCOFFERINC	PMSCOFFERING/PMSCCUSTOMMBOS	PMSCCUSTOMMBOS
IMGLIB	PMSCOFFERINC	PMSCOFFERING/IMGLIB	IMGLIB

Figure 13-9 PMSC_OFFERING object structure relationships

13.3.3 Migration groups

The migration group for the service offering is the PMSC_OFFERING migration group, which is provided for you to duplicate and modify only as necessary.

As shown in Figure 13-10, the object structure order is important. There also are gaps in the numbering order with which you can add any object structures that you require.



Object	Description	Migration Object Order
AUNC	Launch entry migration object for Migration Mai	5
TION	Signature option migration object for Migration I	10
TFLA(	Advanced signature option migration object for	15
JTOS(	PMSC Automation Scripts	20
SSETA	Asset attribute migration object	25
SIFICA	Classifications migration object	30
ACTIC	Action migration object for Migration Manager	35
BPLA	Jobplan migration object for Migration Manager	40
)MMOI	Commodities migration object	45
N	Action migration object for Migration Manager	50
NGRC	Action Group migration object for Migration Mai	55
OCES!	Workflow process definition migration object fc	60
IPLAT	Ticket Template migration object	65
ROUP	Security group migration object	70
FFERIN	SRM 72 Service Offering migration object	75

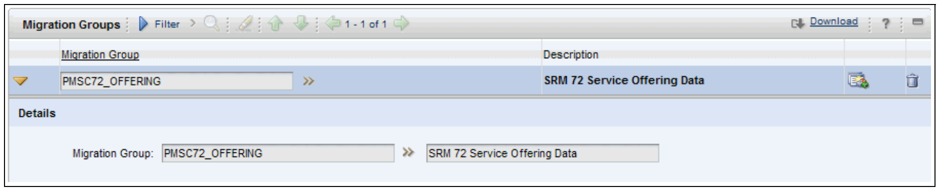
Figure 13-10 Service offering migration group object order

13.3.4 Package definitions

This section describes most of the work that you must perform. The SQL definitions that IBM supplies require you to modify the definition to migrate the client-modified and client-added offerings.

Access the package

Browse to the Migration Manager application by clicking **Go To** → **System Configuration** → **Migration** → **Migration Manager** and then search for `PMSC75_ServiceTemplate`. When the template is found, click the **Package Definition** tab, and then click the icon, as shown in Figure 13-11.

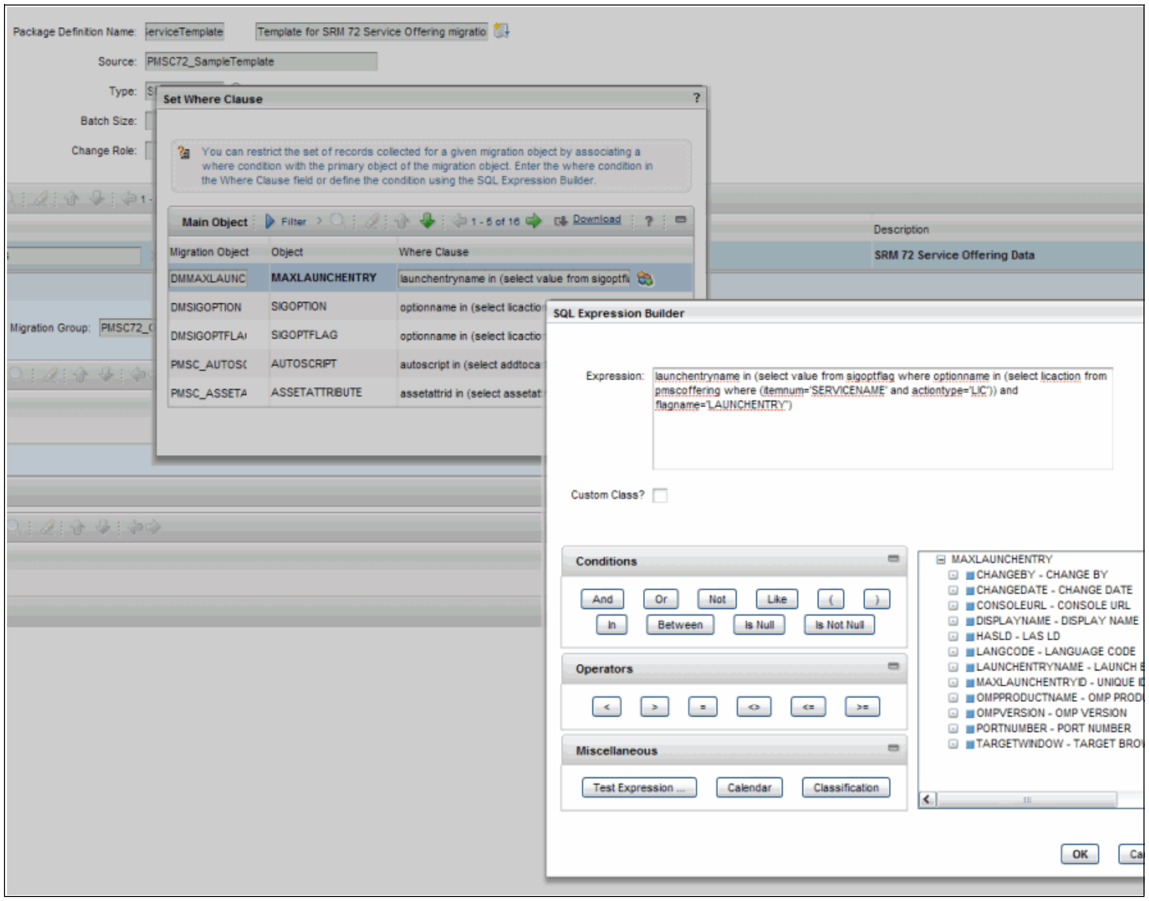


Migration Group	Description
PMSC72_OFFERING	SRM 72 Service Offering Data

Migration Group: PMSC72_OFFERING SRM 72 Service Offering Data

Figure 13-11 Migration package definition

Figure 13-12 shows one of the SQL criteria statements for the various packages.



Package Definition Name: erviceTemplate   Template for SRM 72 Service Offering migratio

Source: PMSC72_SampleTemplate

Set Where Clause

You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

Migration Object	Object	Where Clause
DMMAXLAUNC	MAXLAUNCHENTRY	launchentryname in (select value from sigoptfl
DMSIGOPTION	SIGOPTION	optionname in (select licactio
DMSIGOPTFLA	SIGOPTFLAG	optionname in (select licactio
PMSC_AUTOS	AUTOSCRIPT	autoscript in (select addtoca
PMSC_ASSETA	ASSETATTRIBUTE	assetattrid in (select assetat

Description: SRM 72 Service Offering Data

Migration Group: PMSC72_C

SQL Expression Builder

Expression: launchentryname in (select value from sigoptflag where optionname in (select licaction from pmscoffering where (itemnum='SERVICENAME' and actiontype='LIC')) and flagname='LAUNCHENTRY')

Custom Class? ☐

MAXLAUNCHENTRY
- CHANGEBY - CHANGE BY
- CHANGEDATE - CHANGE DATE
- CONSOLEURL - CONSOLE URL
- DISPLAYNAME - DISPLAY NAME
- HASLD - LAS LD
- LANGCODE - LANGUAGE CODE
- LAUNCHENTRYNAME - LAUNCH E
- MAXLAUNCHENTRYID - UNIQUE I
- OMPPRODUCTNAME - OMP PROD
- OMPVERSION - OMP VERSION
- PORTNUMBER - PORT NUMBER
- TARGETWINDOW - TARGET BRO

Figure 13-12 Set Where clause dialog box with MAXLAUNCHENTRY selected

Defining the SQL definitions

Access the Expression Builder by clicking the expression builder icon on the far right side of each row. Manipulate the SQL for each migration object in the group (see Example 13-1).

Example 13-1 Sample caption

```
launchentryname in (select value from sigoptflag where optionname in
(select licaction from pmscoffering where (itemnum='SERVICENAME' and
actiontype='LIC')) and flagname='LAUNCHENTRY')
```

In this code, replace SERVICENAME with the individual itemnum values or another construct that satisfies your record selection criteria.

By using the SQL syntax in Example 13-2, you can select several offerings to migrate.

Example 13-2 Sample caption

```
launchentryname in (select value from sigoptflag where optionname in
(select licaction from pmscoffering where (itemnum in
('SERVICENAME1','SERVICENAME2','SERVICENAME3','SERVICENAME4') and
actiontype='LIC')) and flagname='LAUNCHENTRY')
```

In this fashion, you choose to migrate four service offerings. This criteria must match the other object structures' criteria to ensure the correct migration.

Example 13-3 shows how to migrate all of the offerings for this specific migration object.

Example 13-3 Sample caption

```
launchentryname in (select value from sigoptflag where optionname in
(select licaction from pmscoffering where (actiontype='LIC')) and
flagname='LAUNCHENTRY')
```

Criteria must match: The selection criteria must match from one object structure to another object structure to ensure that the correct record sets are migrated. The syntax of the SQL and the various artifacts that are used to create the record set might vary, but the criteria must be the same.

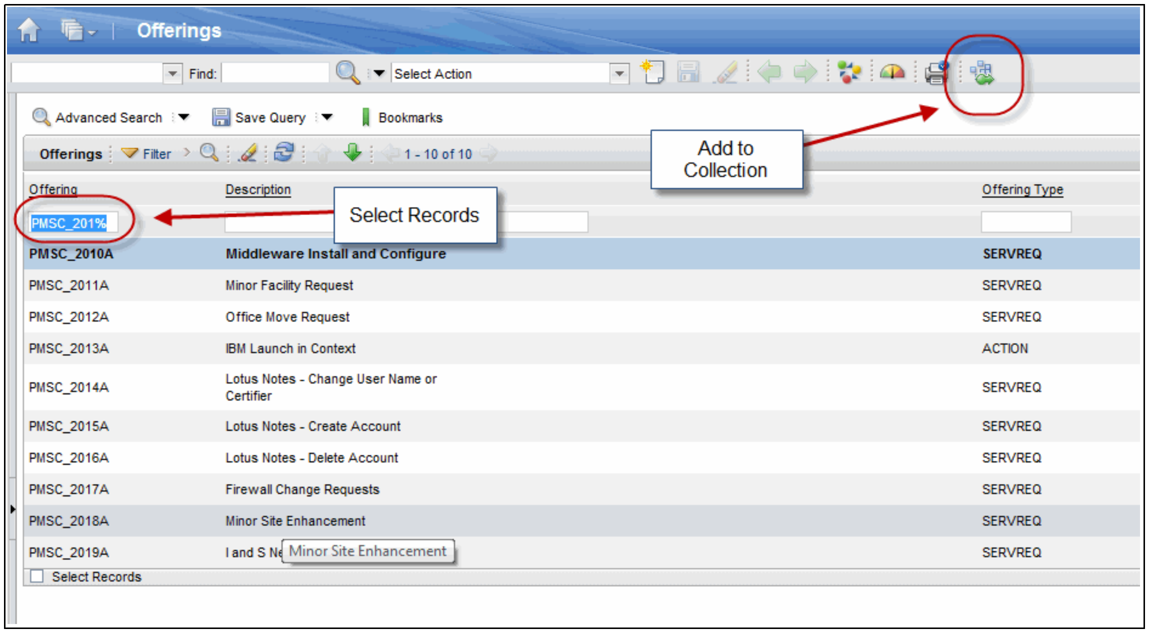
13.4 Solution B

In this approach, you access the Migration Collections application and develop the migration package by selecting **Go To** → **System Configuration** → **Migration** → **Migration Collection**.

13.4.1 Define collection

After you access the Migration Collections application, you create the service offerings collection by using any of the methods that are described in Chapter 3, "Migration collections" on page 49. The key difference for this specific scenario is rather than defining the limitations to the package at the Migration Package level (as defined by the SQL clause), you create the collection with only the specific configurations you want, which can be the same criteria that is used in the application rather than as described in "Defining the SQL definitions" on page 323.

For example, where it now indicates the itemnum in a list of `SERVICENAME` offerings, instead browse to the Offering application and select the offering items by using standard techniques to filter the wanted records. Now, add the configurations by clicking the **Add to migrations collections** option. Figure 13-13 shows the selection and use of this toolbar option.



Offering	Description	Offering Type
PMSC_201%		
PMSC_2010A	**Middleware Install and Configure**	**SERVREQ**
PMSC_2011A	Minor Facility Request	SERVREQ
PMSC_2012A	Office Move Request	SERVREQ
PMSC_2013A	IBM Launch in Context	ACTION
PMSC_2014A	Lotus Notes - Change User Name or Certifier	SERVREQ
PMSC_2015A	Lotus Notes - Create Account	SERVREQ
PMSC_2016A	Lotus Notes - Delete Account	SERVREQ
PMSC_2017A	Firewall Change Requests	SERVREQ
PMSC_2018A	Minor Site Enhancement	SERVREQ
PMSC_2019A	I and S Ne	SERVREQ

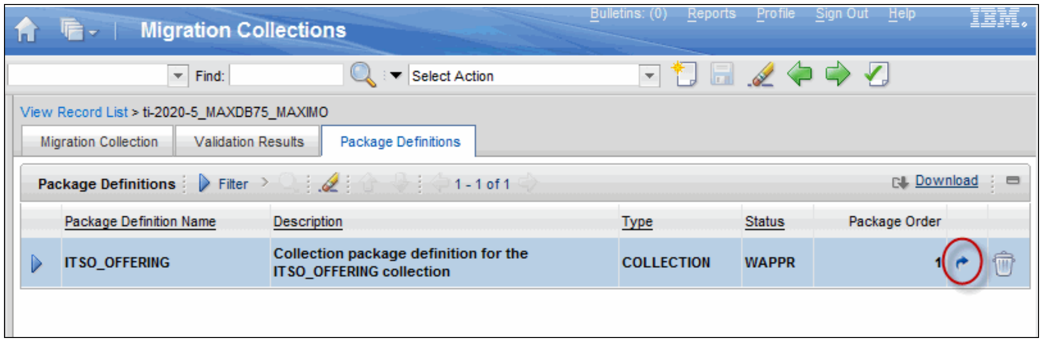
Figure 13-13 Selecting configuration items with list record criteria and toolbar option

13.4.2 Validate collection

After you finish selecting the collection items, click **Select Action → Validate Migration Collection**. After you validate the collection, review the validation configurations. Add those configurations that are required to avoid package definition errors.

Package definition

Click **Select Action → Create Package Definition.** From the package definition tab, you can browse to the Migration manager application for deployment by clicking the **Go to Migration Manager application** icon, as shown in Figure 13-14.



Package Definition Name	Description	Type	Status	Package Order
ITSO_OFFERING	Collection package definition for the ITSO_OFFERING collection	COLLECTION	WAPPR	1

Figure 13-14 Browsing to the Migration Manager application from the Migration Collections application

13.5 Deployment

After the package is created, you must distribute the package to get it to the target environment. Use file distribution. For more information, see *Maximo 7.5 product information for Migration Manager* at this website:

http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/gp_migmgr/c_ctr_mig_mgr_over.html

Migration: The migration of service offering does not require an Admin mode or server restart.

After the deployment to the target environment, users can start to use the service offering in the target environment.

13.5.1 Deployment considerations

The primary considerations for this particular migration package are that you must ascertain that there are changes in the migration groups' artifacts that the content requires for the Migration Manager to migrate the content correctly. You must also ascertain that the changes were made to the content that you must move from one environment to the next environment.

If either of these conditions is not true, do not migrate the content because the content is installed by the IBM product installer when the system is first set up.

Because this material is content-based material, you and your implementation team must carefully consider whether the content is required for unit testing and integration testing for production to assimilate the changes.

If business processes require the content to work properly, this chapter is important for you.

Migrating a service catalog

This chapter describes the best practices that are associated with migrating a service catalog.

The *service catalog* consists of numerous standard database objects, which together form a core component of the SmartCloud Control Desk. Service catalogs are made up of service offerings, which are grouped logically. A typical example of a service catalog is an IT service catalog, which might include server hardware and email account creation service offerings.

The SmartCloud Control Desk solution is delivered with a standard pre-formatted snapshot package definition template that is complete with configurable WHERE clauses to include and exclude data in the service catalog migration.

This chapter includes the following sections:

- Requirements
- Solution
- Configuration applications
- Object structures
- Migration groups
- Package definitions
- Deployment

14.1 Requirements

Service catalogs are defined in a development environment and then promoted to test and production environments. As standard, SmartCloud Control Desk is delivered with a Migration Manager template PMSC75_CatalogTemplate that includes the following migration groups that are required to migrate service catalogs: DATADICTIONARY, PMSC75_OFFERING, and PMSC75_CATALOG.

Important: In addition to the PMSC75_CatalogTemplate Migration Manager template, SmartCloud Control Desk is delivered with the template PMSC72_CatalogTemplate. The template PMSC72_CatalogTemplate should not be used to migrate Service catalogs configuration data. It was designed for the Tivoli Service Request Manager product and not for the SmartCloud Control Desk product.

The predefined packages migrate data that is relevant to the following system configurations:

- Data dictionary
- Launch-in-context
- Signature options
- Actions (job plan)
- Job plans
- Commodities
- Workflow processes
- Ticket templates
- Offerings
- Document information
- Security groups
- Classifications
- Asset attributes
- Automated scripts
- Actions
- Action groups
- Service catalog

14.2 Solution

The PMSC75_CatalogTemplate Package definition does not require any changes to the object structures or migration groups. In the PMSC75_OFFERING and PMSC75_CATALOG migration groups, you are required to configure the predefined WHERE clauses that are associated with the object structures to specify the service catalog (CATALOGNAME) to migrate. For the DATADICTIONARY migration group, you must provide the WHERE clauses for each Migration Object.

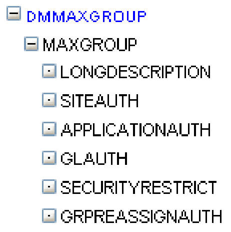
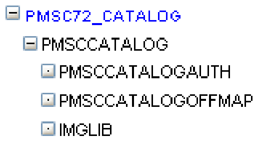
Access the template migration package by browsing to the Migration Manager application by selecting **Go To** → **System Configuration** → **Migration** → **Migration Manager**.

Use the Migration Manager template PMSC75_CatalogTemplate that ships with the product to migrate a service catalog. The PMSC75_CatalogTemplate consists of the DATADICTIONARY, PMSC75_CATALOG, and PMSC75_OFFERING migration groups.

14.3 Configuration applications

In Chapter 13, “Migrating service offering content” on page 311, we described the configuration applications that are associated with an individual service offering (PMSC75_OFFERING migration object). Similarly, the Services Catalog uses the configuration applications that are used to define a service offering with the addition of the Security Groups and Catalogs applications. Table 14-1 shows the other configuration applications and associated object structures.

Table 14-1 Configuration applications

Application name	Object structure
Security Groups	 DMMAXGROUP MAXGROUP LONGDESCRIPTION SITEAUTH APPLICATIONAUTH GLAUTH SECURITYRESTRICT GRPREASSIGNAUTH
Catalogs	 PMSC72_CATALOG PMSCCATALOG PMSCCATALOGAUTH PMSCCATALOGOFFMAP IMGLIB

14.3.1 Security groups

The Security Groups application is accessed from the appropriate product Start Center by selecting **Go To** → **Security** → **Security Groups**.

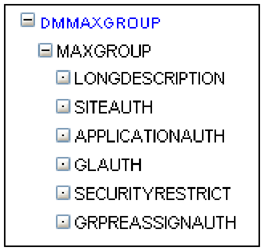
14.3.2 Catalogs

The Catalogs application is accessed from the appropriate product Start Center by selecting **Go To** → **Service Request Catalog** → **Catalogs**.

14.4 Object structures

The following object structures are part of the PMSC75_CatalogTemplate package definition that ships with the product.

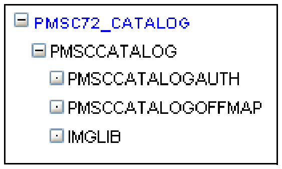
As part of the PMSC75_CATALOG migration group, the DMMAXGROUP object structure supports the service catalog migration. Figure 14-1 shows the contents of the DMMAXGROUP object structure.



```
DMMAXGROUP
  MAXGROUP
    LONGDESCRIPTION
    SITEAUTH
    APPLICATIONAUTH
    GLAUTH
    SECURITYRESTRICT
    GRPREASSIGNAUTH
```

Figure 14-1 DMMAXGROUP object structure

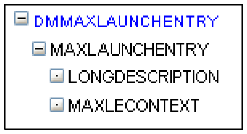
As part of the PMSC75_CATALOG migration group, the PMSC72_CATALOG object structure supports the service catalog migration. Figure 14-2 shows the contents of the PMSC72_CATALOG object structure.



```
PMSC72_CATALOG
  PMSCCATALOG
    PMSCCATALOGAUTH
    PMSCCATALOGOFFMAP
    IMGLIB
```

Figure 14-2 PMSC72_CATALOG object structure

As part of the PMSC75_OFFERING migration group, the DMMAXLAUNCHENTRY object structure supports the service catalog migration. Figure 14-3 shows the contents of the DMMAXLAUNCHENTRY object structure.



```
DMMAXLAUNCHENTRY
  MAXLAUNCHENTRY
    LONGDESCRIPTION
    MAXLECONTEXT
```

Figure 14-3 DMMAXLAUNCHENTRY object structure

As part of the PMSC75_OFFERING migration group, the DMSIGOPTION object structure supports the service catalog migration. Figure 14-4 shows the contents of the DMSIGOPTION object structure.

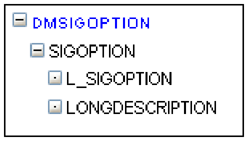


Figure 14-4   DMSIGOPTION object structure

As part of the PMSC75_OFFERING migration group, the DMSIGOPTFLAG object structure supports the service catalog migration. Figure 14-5 shows the contents of the DMSIGOPTFLAG object structure.

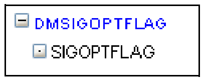


Figure 14-5   DMSIGOPTFLAG object structure

As part of the PMSC75_OFFERING migration group, the PMSC_JPACTION object structure supports the service catalog migration. Figure 14-6 shows the contents of the PMSC_JPACTION object structure.

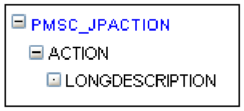
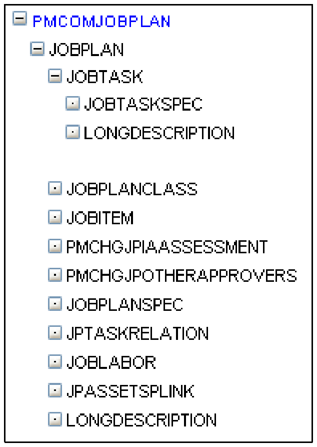


Figure 14-6   PMSC_JPACTION object structure

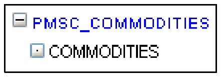
As part of the PMSC75_OFFERING migration group, the PMCOMJOBPLAN object structure supports the service catalog migration. Figure 14-7 shows the contents of the PMCOMJOBPLAN object structure.



```
PMCOMJOBPLAN
  JOBPLAN
    JOBTASK
      JOBTASKSPEC
      LONGDESCRIPTION

    JOBPLANCLASS
    JOBITEM
    PMCHGJPIAASSESSMENT
    PMCHGJPOTHERAPPROVERS
    JOBPLANSPEC
    JPTASKRELATION
    JOBLABOR
    JPASSETSPLINK
    LONGDESCRIPTION
```

Figure 14-7 PMCOMJOBPLAN object structure

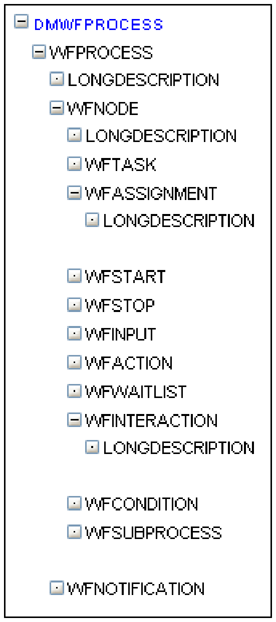
As part of the PMSC75_OFFERING migration group, the PMSC_COMMODITIES object structure supports service catalog migration. Figure 14-8 shows the contents of the PMSC_COMMODITIES object structure.



```
PMSC_COMMODITIES
  COMMODITIES
```

Figure 14-8 PMSC_COMMODITIES object structure

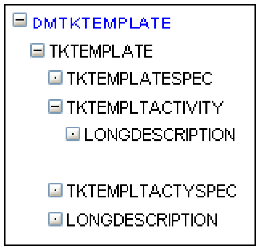
As part of the PMSC75_OFFERING migration group, the DMWFPROCESS object structure supports the service catalog migration. Figure 14-9 shows the contents of the DMWFPROCESS object structure.



```
⊟ DMWFPROCESS
  ⊟ WFPROCESS
    ⊡ LONGDESCRIPTION
    ⊟ WFNODE
      ⊡ LONGDESCRIPTION
      ⊡ WFTASK
      ⊟ WFASSIGNMENT
        ⊡ LONGDESCRIPTION

      ⊡ WFSTART
      ⊡ WFSTOP
      ⊡ WFINPUT
      ⊡ WFACTION
      ⊡ WFWAITLIST
      ⊟ WFINTERACTION
        ⊡ LONGDESCRIPTION

      ⊡ WFCONDITION
      ⊡ WFSUBPROCESS

    ⊡ WFNOTIFICATION
```

Figure 14-9 DMWFPROCESS object structure

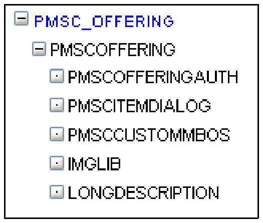
As part of the PMSC75_OFFERING migration group, the DMTKTEMPLATE object structure supports the service catalog migration. Figure 14-10 shows the contents of the DMTKTEMPLATE object structure.



```
⊟ DMTKTEMPLATE
  ⊟ TKTEMPLATE
    ⊡ TKTEMPLATESPEC
    ⊟ TKTEMPLTACTIVITY
      ⊡ LONGDESCRIPTION

    ⊡ TKTEMPLTACTYSPEC
    ⊡ LONGDESCRIPTION
```

Figure 14-10 DMTKTEMPLATE object structure

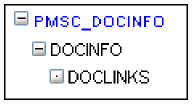
As part of the PMSC75_OFFERING migration group, the PMSC_OFFERING object structure supports the service catalog migration. Figure 14-11 shows the contents of the PMSC_OFFERING object structure.



```
PMSC_OFFERING
  PMSCOFFERING
    PMSCOFFERINGAUTH
    PMSCITEMDIALOG
    PMSCCUSTOMMBOS
    IMGLIB
    LONGDESCRIPTION
```

Figure 14-11 PMSC_OFFERING object structure

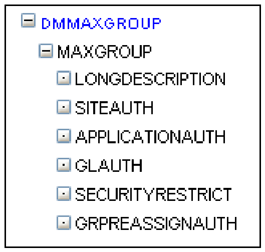
As part of the PMSC75_OFFERING migration group, the PMSC_DOCINFO object structure supports the service catalog migration. Figure 14-12 shows the contents of the PMSC_DOCINFO object structure.



```
PMSC_DOCINFO
  DOCINFO
    DOCLINKS
```

Figure 14-12 PMSC_DOCINFO object structure

As part of the PMSC75_OFFERING migration group, the DMMAXGROUP object structure supports the service catalog migration. Figure 14-13 shows the contents of the DMMAXGROUP object structure.



```
DMMAXGROUP
  MAXGROUP
    LONGDESCRIPTION
    SITEAUTH
    APPLICATIONAUTH
    GLAUTH
    SECURITYRESTRICT
    GRPREASSIGNAUTH
```

Figure 14-13 DMMAXGROUP object structure

As part of the PMSC75_OFFERING migration group, the DMCLASSIFICATION object structure supports the service catalog migration. Figure 14-14 shows the contents of the DMCLASSIFICATION object structure.

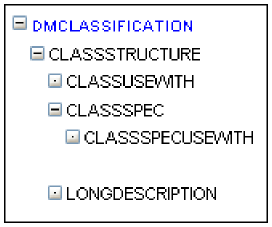


Figure 14-14 DMCLASSIFICATION object structure

As part of the PMSC75_OFFERING migration group, the PMSC_ASSETATTRIBUTE object structure supports the service catalog migration. Figure 14-15 shows the contents of the PMSC_ASSETATTRIBUTE object structure.

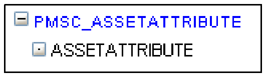


Figure 14-15 PMSC_ASSETATTRIBUTE object structure

As part of the PMSC75_OFFERING migration group, the DMSCRIPT object structure supports the service catalog migration. Figure 14-16 shows the contents of the DMSCRIPT object structure.

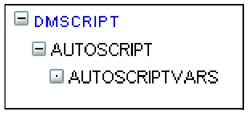


Figure 14-16 DMSCRIPT object structure

As part of the PMSC75_OFFERING migration group, the DMACTION object structure supports the service catalog migration. Figure 14-17 shows the contents of the DMACTION object structure.

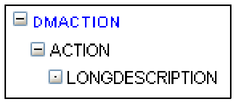


Figure 14-17 DMACTION object structure

As part of the PMSC75_OFFERING migration group, the DMACTIONGROUP object structure supports the service catalog migration. Figure 14-18 shows the contents of the DMACTIONGROUP object structure.

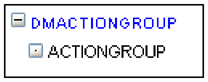


Figure 14-18 DMACTIONGROUP object structure

You are not required to create more object structures or modify the existing object structures to migrate a service catalog.

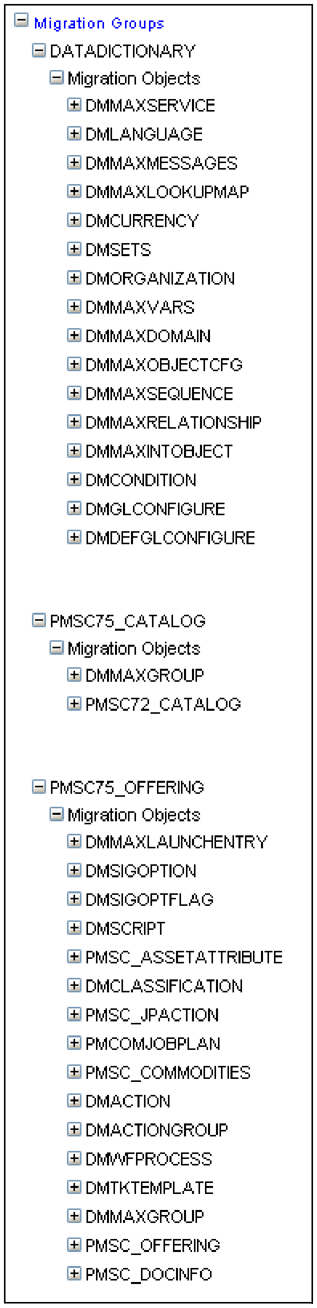
In the previous scenario, we described the special features of the PMSC_OFFERING object structure. These features also are applicable to the service catalog migration scenario.

14.5 Migration groups

This section describes the migration groups that ship with the product and that migrate a designated service catalog. The object structures that support service catalog migration belong to the following migration groups:

- DATADICTIONARY
- PMSC75_OFFERING
- PMSC75_CATALOG

Figure 14-19 on page 340 shows the predefined DATADICTIONARY, PMSC75_OFFERING, and PMSC75_CATALOG migration groups and their dependencies.



```
⊟ Migration Groups
  ⊟ DATADICTIONARY
    ⊟ Migration Objects
      ⊞ DMMAXSERVICE
      ⊞ DMLANGUAGE
      ⊞ DMMAXMESSAGES
      ⊞ DMMAXLOOKUPMAP
      ⊞ DMCURRENCY
      ⊞ DMSETS
      ⊞ DMORGANIZATION
      ⊞ DMMAXVARS
      ⊞ DMMAXDOMAIN
      ⊞ DMMAXOBJECTCFG
      ⊞ DMMAXSEQUENCE
      ⊞ DMMAXRELATIONSHIP
      ⊞ DMMAXINTOBJECT
      ⊞ DMCONDITION
      ⊞ DMGLCONFIGURE
      ⊞ DMDEFGLCONFIGURE

  ⊟ PMSC75_CATALOG
    ⊟ Migration Objects
      ⊞ DMMAXGROUP
      ⊞ PMSC72_CATALOG

  ⊟ PMSC75_OFFERING
    ⊟ Migration Objects
      ⊞ DMMAXLAUNCHENTRY
      ⊞ DMSIGOPTION
      ⊞ DMSIGOPTFLAG
      ⊞ DMSCRIPT
      ⊞ PMSC_ASSETATTRIBUTE
      ⊞ DMCLASSIFICATION
      ⊞ PMSC_JPACTION
      ⊞ PMCOMJOBPLAN
      ⊞ PMSC_COMMODITIES
      ⊞ DMACTION
      ⊞ DMACTIONGROUP
      ⊞ DMWFPROCESS
      ⊞ DMTKTEMPLATE
      ⊞ DMMAXGROUP
      ⊞ PMSC_OFFERING
      ⊞ PMSC_DOCINFO
```

Figure 14-19 Migration groups structure

14.6 Package definitions

This section describes most of the work that must be done to migrate a service catalog from a Source to a target environment. To migrate a designated service catalog, duplicate the Migration Manager template PMSC75_CatalogTemplate, then, in each migration group, modify the WHERE clauses that are provided with the template by IBM.

To access the package, browse to the Migration Manager application by selecting **Go To** → **System Configuration** → **Migration** → **Migration Manager**. Search for and select **PMSC75_CatalogTemplate**.

In the following sections, we describe the WHERE clauses that are supplied with the template and how to modify them.

14.6.1 DATADICTIONARY

The DATADICTIONARY migration group is delivered with standard WHERE clauses for several object structures in the migration group. You must edit these WHERE clauses to specify the name of the catalog to migrate. You insert the Catalog Name (as it appears in the database) in each WHERE clause where CATALOGNAME appears. For example, to migrate the SERVICE CATALOG1 service catalog, WHERE clause A (as shown in Example 14-1) is replaced with WHERE clause B (as shown in Example 14-2).

Example 14-1 WHERE clause A

```
objectname in (select mboname from pmsccustommbos where itemnum in
(select offeringnum from pmsccatalogoffmap where
itemnum='CATALOGNAME'))
```

Example 14-2 WHERE clause B

```
objectname in (select mboname from pmsccustommbos where itemnum in
(select offeringnum from pmsccatalogoffmap where itemnum='SERVICE
CATALOG1'))
```

Important: In the DATADICTIONARY Migration Group, set the WHERE clauses to 1=2 for the following Migration Objects that do not need to be migrated: DMMAXSERVICE, DMLANGUAGE, DMMAXMESSAGES, DMCURRENCY, DMSETS, DMORGANIZATION, DMMAXVARS, DMCONDITION, DMDEFGLCONFIGURE, DMGLCONFIGURE.

Table 14-2 lists the predefined DATADICTIONARY WHERE clauses, their respective object structures, and the order in which they are migrated.

Table 14-2 Predefined DATADICTIONARY WHERE clauses

Migration order	Object structure	WHERE clause
9	DMMAXDOMAIN	domainid in (select domainid from maxattributecfg where objectname in (select mboname from pmsccustommbos where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))
10	DMMAXOBJECTCFG	objectname in (select mboname from pmsccustommbos where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
11	DMMAXSEQUENCE	tbname in (select mboname from pmsccustommbos where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
12	DMMAXLOOKUPMAP	target in (select mboname from pmsccustommbos where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
13	DMMAXRELATIONSHIP	parent = 'SR' and child in (select mboname from pmsccustommbos where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
14	DMMAXINTOBJECT	intobjectname in (select intobjectname from maxintobjdetail where objectname in (select mboname from pmsccustommbos where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))

14.6.2 PMSC75_OFFERING

The PMSC75_OFFERING migration group is delivered with standard WHERE clauses for each object structure in the migration group. As with the DATADICTIONARY migration group, you must modify the WHERE clauses and specify the name of the service catalog to migrate.

Table 14-3 lists the object structures that make up the PMSC75_OFFERING migration object and the associated WHERE clauses in the order in which they are migrated.

Table 14-3 Predefined PMSC75_OFFERING WHERE clauses

Migration order	Object structure	WHERE clause
5	DMMAXLAUNCH	launchentryname in (select value from sigoptflag where optionname in (select licaction from pmscoffering where actiontype='LIC' and itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) and flagname='LAUNCHENTRY')
10	DMSIGOPTION	optionname in (select licaction from pmscoffering where actiontype='LIC' and itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
15	DMSIGOPTIONFLAG	optionname in (select licaction from pmscoffering where actiontype='LIC' and itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) and flagname='LAUNCHENTRY'
20	DMSCRIPT	autoscript in (select addtocartscript from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) or autoscript in (select offsubmitscript from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) or autoscript in (select prepopulationscript from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) or autoscript in (select validationattr from pmscoffdialog where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
25	PMSC_ASSETATTRIBUTE	assetattrid in (select assetattrid from pmscitemspec where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))

Migration order	Object structure	WHERE clause
30	DMCLASSIFICATION	classstructureid in (select classstructureid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) or classstructureid in (select classstructureid from pmsccatalogoffmap where itemnum='CATALOGNAME') or classstructureid in (select classstructureid from tktemplate where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))) or classstructureid in (select classstructureid from tktempltactivity where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))) or classstructureid in (select classstructureid from jobplan where jpnum in (select jpnum from tktempltactivity where templateid in (select templateid from tktemplate where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))))) or classstructureid in (select classstructureid from jobtask where jpnum in (select jpnum from tktempltactivity where templateid in (select templateid from tktemplate where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))))
35	PMSC_JPACTION	action in (select flowaction from jobplan where jpnum in (select jpnum from tktempltactivity where templateid in (select templateid from tktemplate where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))))) or action in (select flowaction from jobtask where jpnum in (select jpnum from tktempltactivity where templateid in (select templateid from tktemplate where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))))

Migration order	Object structure	WHERE clause
40	PMCOMJOBPLAN	(jpnum in (select jpnum from tktempltactivity where templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))) or jpnum in (select jpnum from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))) and status in (select value from synonymdomain where domainid='JOBPLANSTATUS' and maxvalue='ACTIVE')
45	PMSC_COMMODITIES	commodity in (select commodity from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) or commodity in (select commoditygroup from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
50	DMACTION	action in (select action from wfaction where processname in ((select mgrapprprocess from pmscofferingext where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) union (select srapprprocess from pmscofferingext where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))) or action in (select member from actiongroup where action in (select action from wfaction where processname in ((select mgrapprprocess from pmscofferingext where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) union (select srapprprocess from pmscofferingext where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))))
55	DMACTIONGROUP	action in (select action from wfaction where processname in ((select mgrapprprocess from pmscofferingext where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) union (select srapprprocess from pmscofferingext where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))))

Migration order	Object structure	WHERE clause
60	DMWFPROCESS	processname in (select srapprprocess from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')) or processname in (select mgrapprprocess from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
65	DKTICKETTEMPLATE	templateid in (select templateid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
70	DMMAXGROUP	groupname in (select groupname from pmscofferingauth where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME'))
75	PMSC_OFFERING	itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')
80	PMSC_DOCINFO	docinfoid in (select docinfoid from doclinks where ownertable='PMSCOFFERING' and ownerid in (select itemid from pmscoffering where itemnum in (select offeringnum from pmsccatalogoffmap where itemnum='CATALOGNAME')))

Important: The migration order values increase in increments of five to allow for more object structures to be added between the predefined object structures in the future.

14.6.3 PMSC75_CATALOG

The PMSC75_CATALOG migration group is delivered with standard WHERE clauses for each object structure in the migration group. As with the PMSC75_OFFERING migration group, you must edit these WHERE clauses to specify the name of the service catalog to migrate. Insert the catalog name, as it appears in the database, in each WHERE clause.

Table 14-4 lists the object structures that make up the PMSC75_CATALOG migration group and the associated WHERE clause in the order in which they are migrated.

Table 14-4 Predefined PMSC75_CATALOG WHERE clauses

Migration order	Object structure	WHERE clause
5	DMMAXGROUP	groupname in (select groupname from pmsccatalogauth where itemnum='CATALOGNAME')
10	PMSC72_CATALOG	itemnum='CATALOGNAME'

14.7 Deployment

You must understand the deployment considerations that are associated with this scenario before the package to the target environment is deployed; in particular, the migration prerequisites.

After the package definition is approved and activated, you can create and distribute the package. Then, deploy the package to the target environment.

Resource: This publication does not describe the standard steps that are involved in deploying a package. For more information, see this website:

http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/index.jsp?topic=%2Fcom.ibm.mbs.doc%2Fgp_migmgr%2Ft_ctr_deploy_packages.html

After deployment to the target environment, users can start to use the service catalog and its related service offerings in the target environment.

14.7.1 Deployment considerations

You must accommodate for the following specified deployment considerations before the deployment of the PMSC75_CatalogOffering, or the package migration fails.

Migration prerequisites

The PMSC75_CatalogTemplate migrates the other records that are directly associated with the service catalog. The package definition that ships with the product is not an all encompassing package to migrate everything in Maximo. Therefore, it is important to ensure that the following objects are migrated manually or by using alternative migration packages:

- Job plans

  The PMSC75_OFFERING migration group migrates the job plans in their entirety (with the exclusion of nested job plans); however, it does not create the referenced labor, items, services, and tools in their respective database objects in the target environment.

- Nested job plans

  Where a job plan is specified against a job plan task (nested job plan), the nested job plan is not migrated as part of the PMSC75_CatalogTemplate. Only the job plan that is associated directly with the ticket template that is applied to the service catalog is migrated in the package. The nested job plan must be migrated in advance by using Migration Manager Snapshot or Collection package.

- Response plans

  Response plans are not included in the predefined service catalog migration package. Response plans that are associated with service groups and service offerings must be migrated before the PMSC75_CatalogTemplate is deployed to the target environment.

- Parent classifications

  To migrate a classification hierarchy, you must modify the DMCLASSIFICATION Migration Object in the PMSC75_OFFERING migration group. For more information about classifications migration, see Chapter 10, “Migrating classifications” on page 253.

- Circular relationships

  A situation can occur where actions reference workflows, and workflows reference actions. The PMSC75_OFFERING migrates the first level of actions that are associated with relevant workflows. You must first migrate workflows that are referenced by actions, and the actions that are referenced by workflows before migrating service catalogs. For more information about workflows migration, see Chapter 9, “Migrating workflows” on page 235.

WHERE clause inclusions and exclusions

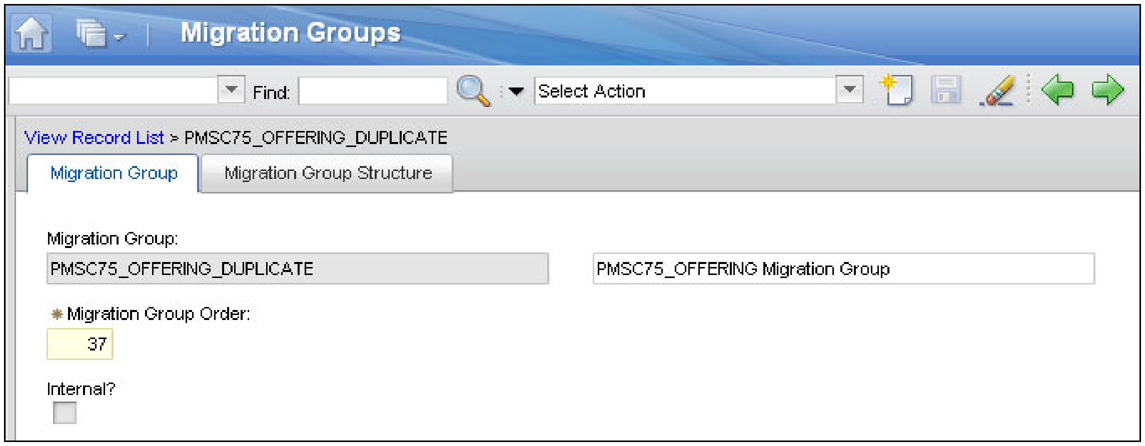
WHERE clauses include and exclude data from the migration package. The PMSC75_CatalogTemplate migrates objects only where they are relevant to the CATALOGNAME that is specified in each of the WHERE clauses that is associated with each of the object structures. For example, if a classification is not specified against the job plan, an offering of a catalog, or against the catalog, the migration group does not migrate any of the objects in the DMCLASSIFICATION object structure.

Duplicating the PMSC75_OFFERING migration group

In the PMSC75_CatalogTemplate, the migration order of the PMSC75_OFFERING Migration Group is lower than the order of the PMSC75_CATALOG migration group. To migrate the package correctly, the PMSC75_OFFERING Migration Group must be processed before the PMSC75_CATALOG Migration Group.

It might be necessary to modify the predefined PMSC75_OFFERING migration group to meet specific migration needs. However, because the migration group also is used in the PMSC75_ServiceTemplate, we advise that you duplicate it. After it is duplicated, you must be aware that when added to the PMSC75_CatalogTemplate, the auto-number on the migration groups sets the duplicated PMSC75_OFFERING to a higher number than the PMSC75_CATALOG. Therefore, the PMSC75_CATALOG processes before the duplicated PMSC75_OFFERING migration group. In this instance, the package fails if the prerequisite data for the predefined original PMSC75_OFFERING migration group does not exist in the target environment.

Important: To avoid the migration failure issue, after the PMSC75_OFFERING migration group is duplicated, the Migration Group Order field on the PMSC75_CATALOG migration group must be set to a number greater than the number of the duplicated PMSC75_OFFERING migration group. Figure 14-20 on page 350 shows the Migration Group Order field that is automatically populated on the duplication of the PMSC75_OFFERING migration group.


Migration Groups
Find:
Select Action
View Record List > PMSC75_OFFERING_DUPLICATE
Migration Group
Migration Group Structure

Migration Group:
PMSC75_OFFERING_DUPLICATE
PMSC75_OFFERING Migration Group

Migration Group Order:
37

Internal?

Figure 14-20 Migration Group Order field in the Migration Groups application

15 Common topics

The Migration Manager has many built-in capabilities, most of which are used in almost every migration scenario. You also must be aware of certain considerations.

This chapter includes the following sections:

- Comparing Integration Framework and Migration Manager
- Multiple language considerations
- Snapshot package versus Change package
- Embedded URLs
- Clustered environment considerations
- Change tracking and ad hoc reporting
- Admin mode
- Migrating hierarchical data
- Migration Manager comparisons
- Error correction and package reprocessing
- Setting up logging for the Migration Manager
- Start Center visibility into configurations

15.1 Comparing Integration Framework and Migration Manager

Integration Framework and the Migration Manager are two key components of Tivoli's process automation engine. These components have various export and import capabilities. Each component is designed for separate purposes. Table 15-1 compares the key features of the two components.

Table 15-1 Comparing Integration Framework and Migration Manager

	Integration Framework	**Migration Manager**
Purpose	► Transactional integration to external applications ► Export and import to and from external applications or existing applications ► Service invocations	Package-based configuration export and import between similar production environments.
Process/ discipline	Enterprise application integration and service-oriented architecture.	Change management
Primary content support	Master and transactional data	Metadata (configuration data)
Other content	Any business object potentially can be targeted for integration.	Any business object potentially can be targeted for migration.
Unit of work	Transaction, file, or message (can be a single document or record).	Package (multiple records)
Typical user	Practitioner or implementer with programming skills, experience with application integration, familiarity with ERP systems.	Practitioner or implementer with product configuration skills and familiarity with change management.
Framework	Integration Framework that is built from the ground up.	Migration framework that is built from the ground up; uses several integration constructs.

	Integration Framework	Migration Manager
Key feature differences	Extensive mapping and transformation capabilities that are based on Java, XSL, or rules.	No mapping or transformation capabilities; however, offers value replacement functionality.
	Database event-based single transactions from an application.	Database event-based change tracking and aggregation across multiple applications (change package type or migration collections).
	Asynchronous processing by using Java Message Service (JMS) queues.	No queues
	Tivoli's process automation engine-attached document can be extracted and processed with the parent record.	Tivoli's process automation engine-attached document can be extracted and processed with the parent record; arbitrary attachments (Java class or image files) can be placed in a package.
	N/A	Automation of database configuration during package deployment.
	Expose web services - Web Services Description Language (WSDL) or Representational State Transfer (REST)-based.	N/A
	Start web services	N/A
	Extensive error correction and resubmission capability at the transaction level (Message Reprocessing application).	Error correction and resubmission capability of individual configurations within a package (Deployment Data Errors feature).
	Single object structure processing (data that is exported for multiple files to be sequenced manually for import).	► Multiple object structures are grouped into a single migration group ► Every object structure in a group is sequenced, precluding the need for manual sequencing ► Migration groups are also sequenced
	XML file that contains data can be edited and associated with action attributes that determine the action to be performed with the data elements.	Package contains multiple XML files, which are not intended to be edited; action attributes are automatically set up by the Migration Manager.
	N/A	Granular collection of individual configurations over time aligning with development effort (Migration Collections capabilities).

	Integration Framework	Migration Manager
	Import of data from files can be previewed; preview results are per record.	Deployment of a package can be previewed and rollback points that are specified to manage memory consumption. Preview results are per package that is grouped by individual system configuration.

15.1.1 Selecting Integration Framework or Migration Manager

In this section, we review a few key scenarios and identify the appropriate component to use to run the required tasks:

- Scenario 1

  A developer implemented an Information Technology Infrastructure Library (ITIL)-based incident management workflow by using Tivoli's process automation engine Workflow Designer and related applications. This workflow must be promoted to the production environment.

  The Migration Manager is the tool of choice to package up workflow and related configuration data. A single package can contain your workflow and any actions, roles, and communication templates with any custom code that a client must build into a Maximo Enterpirse Archive (EAR).

- Scenario 2

  An existing ticket management application must be integrated with the Tivoli Service Request Manager product. Historical tickets must be loaded into the Tivoli product. Also, the existing ticket management application is used until Tivoli Service Request Manager goes into production.

  Integration Framework (Maximo Enterprise Adapter) is the tool of choice. Existing data, such as tickets, can be loaded into Tivoli Service Request Manager by using a number of methods, including flat files, interface tables, or XML. Periodically, data can be exported out of the existing tool in the form of a file and dropped into a designated folder that Integration Framework can automatically process into Tivoli Service Request Manager.

- Scenario 3

  The client implemented several SAP R/3 modules, including Materials Management. There is a requirement to integrate specific SAP modules to Maximo Asset Management to achieve the total integration of the purchasing and inventory processes.

Integration Framework (Maximo Enterprise Adapter) and the SAP R/3 Adapter are the tools of choice. The SAP R/3 Adapter offers pre-built integration points to SAP R/3 from Maximo. In addition, custom integration points can be built.

- Scenario 4

  The client has a development environment on which a team of developers configures the IBM Tivoli Change and Configuration Management Database product. Configuration includes the addition of new objects (tables), attributes (columns), and domains. Also, the client wants to add new tabs and dialogs to the existing configuration item (CI) application. All of these configurations must be migrated to a User-Acceptance Test (UAT) environment before they are promoted into production.

  The Migration Manager is the tool of choice. It is designed specifically to cater to a controlled promotion of configurations to production from development through UAT. The tool also automates the structural changes to the underlying database that are required as a result of adding new objects and attributes. Therefore, it provides a seamless deployment of a package that contains variable content.

- Scenario 5

  The client must migrate *foundation* data, which is also known as *implementation data,* from development to production to avoid having to reenter data (such as locations and classifications) in the production environment.

  Typically, foundation data consists of discrete sets of data that do not have multiple or deep relationships with other data. The Migration Manager can be used to migrate such foundation data. If an object structure does not exist that supports this type of data, the practitioner or implementer must create an object structure and, if necessary, author Java based processing code. If the object structure is already available from the Integration Framework, it can be duplicated and used with Migration Manager. With its enhanced deployment functionality, including value replacement and error correction, Migration Manager is the tool of choice.

15.2 Multiple language considerations

Tivoli's process automation engine-based products can support multiple languages from a single environment and database. The language configuration for a production environment is set up during product installation and stored in the underlying production database. Every production environment is associated with a base language, and more languages can be enabled as needed by installing the appropriate language packs. As each language is installed, a set of translated content is added into the production database in dedicated multiple language tables.

The enablement of multiple languages is configured for the attribute of a business object. If a particular attribute is enabled for multi-language, a dedicated multi-language table is created to hold the multi-language content for the targeted attribute. The base table always holds the base language content. A number of attributes that ship with the product already are enabled for multi-language. By using these attributes, key functionality in the product is available and presented in the local language that is chosen by the user when other languages are installed and enabled.

For more information about the product's multiple language capabilities and configuration, see the following documentation for Maximo Asset Management 7.1 and 7.5 at these websites:

- `http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.srm.doc_7.1/reference/mam71_sys_admin_guide.pdf`
- `http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/pdfs/pdf_mbs_sysadmin.pdf`

The Migration Manager requires that the base language in the Source and Target production environments is the same. The Migration Manager issues an error message, BMXAA5651E, and prevents the deployment of packages when the base language of a target environment differs from the language of the source environment.

The multi-language configurations of the production environment must be supported from the migration perspective. The following types of tables must be supported when multi-language content is migrated:

- LONGDESCRIPTION
- L_ Multi-language table

Each Migration Manager object structure that is supplied with the product is designed to include the multi-language table if the multi-language table also was shipped with the product. Figure 15-1 shows the DMMAXAPPS object structure that contains the LONGDESCRIPTION and the multi-language tables.

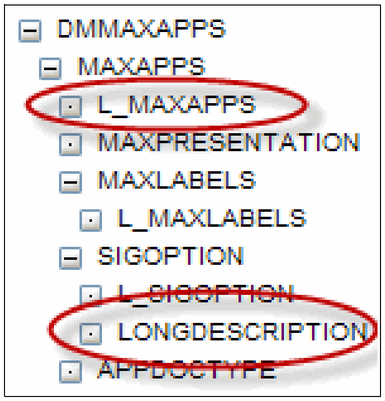


Figure 15-1 DMMAXAPPS object

When configurations are migrated from production environments that have other languages that are enabled, the base language content is extracted by the Migration Manager to be part of the base table. The other language content is extracted as part of the multi-language table. Thus, the Migration Manager automatically exports multi-language-enabled content when a package is created.

Multi-language content: The Migration Manager extracts all of the available multi-language content from the L_ table. There is no capability to extract multi-language content for only certain languages.

This approach to the Migration Manager object structures is especially important when configurations are migrated that constitute the most visible parts of a business application. The various visible elements of an application are configured by using the Application Designer.

When application presentations are migrated, the localized content of various elements of the application presentation also must be migrated. Table 15-2 lists these presentation elements and the corresponding Migration Manager object structures that are enabled to capture multi-language content.

Table 15-2 Presentation elements and the corresponding Migration Manager object structures

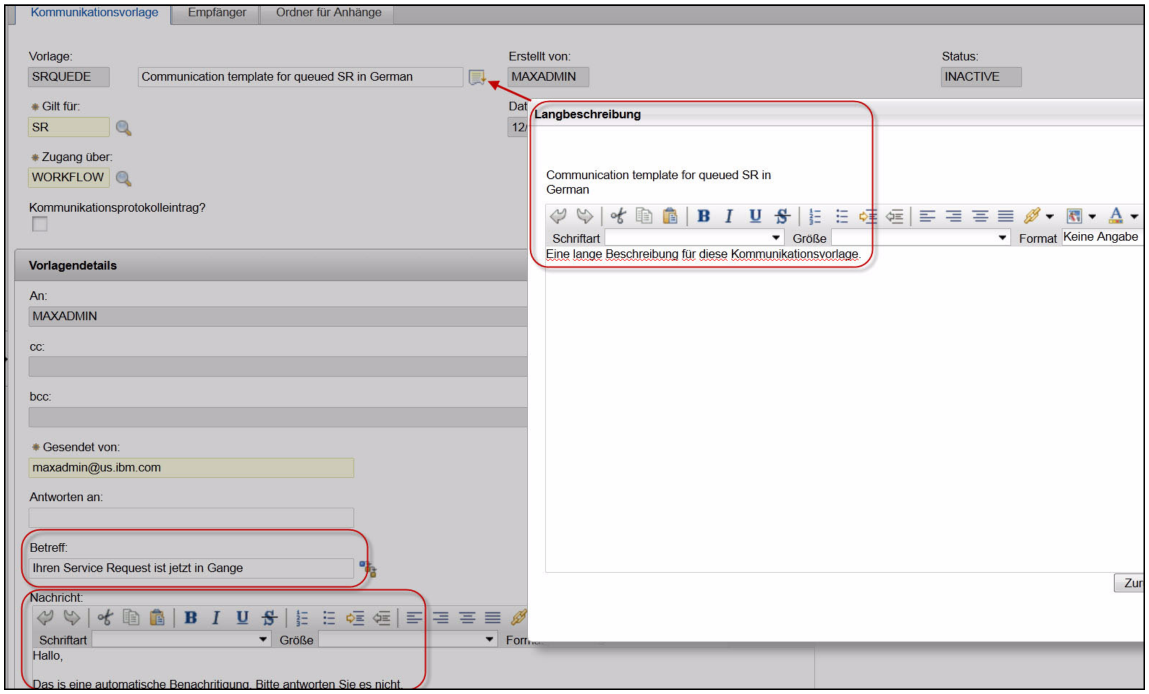
User interface element	**Migration Manager object structure**	**Migration Group**
Presentation XML	DMMAXAPPS	APPLICATION
Labels	Part of DMMAXAPPS	APPLICATION
All menus	DMMAXMENU	APPLICATION
Signature options	DMSIGOPTION	APPSECURITY
Conditional user interface	DMCTRLGROUP	APPSECURITY
Domains	DMMAXDOMAIN	DATADICTIONARY

15.2.1 Base and other language treatment

With many implementations, a base language is chosen to apply to the product environment, but other languages are enabled. The language packs for those languages are applied to the same product environment. A subset of users can log in to the product and chose one of the other languages at login. Migration Manager packages that are created from such non-base language user sessions extract the multi-language data that is associated with a system configuration in a specific manner.

Migration Manager behavior that extracts multi-language is explained by using a scenario. In this scenario, the product environment's base language is English (language code EN). However, other languages are enabled and the corresponding language packs are applied to that product environment. One of the other languages is German (language code DE). Some administrative users use the product and chose German as the language at login time. A requirement is to migrate English and German content from the source environment to appropriate target environments.

One of the administrators creates a Communication Template that contains subject, message, and long description values that are authored in German. Figure 15-2 shows a sample Communication Template with German field values highlighted for the subject (Betreff), message (Nachricht), and long description (Langbeschreibung) fields.



Vorlage: SRQUEDE Communication template for queued SR in German

Erstellt von: MAXADMIN

Status: INACTIVE

Gilt für: SR

Zugang über: WORKFLOW

Kommunikationsprotokolleintrag?

Langbeschreibung

Communication template for queued SR in German

Eine lange Beschreibung für diese Kommunikationsvorlage.

Vorlagendetails

An: MAXADMIN

cc:

bcc:

Gesendet von: maxadmin@us.ibm.com

Antworten an:

Betreff: Ihren Service Request ist jetzt in Gange

Nachricht:

Hallo,

Das is eine automatische Benachrichtigung. Bitte antworten Sie es nicht.

Figure 15-2 Sample Communication Template with German field values

These values are stored as multi-language records in the underlying database. The tables and columns in which they are stored are shown in Table 10-3.

Table 15-3 Multi-language values storage for a communication template

Field	Database Table	Database Column
Subject (Betreff)	L_COMMTEMPLATE	SUBJECT
Message (Nachricht)	L_COMMTEMPLATE	MESSAGE
Long description (Langbeschreibung)	LONGDESCRIPTION	LDTEXT

Because any multi-language table and the long description table must store records with values in multiple languages, the LANGCODE column is used to designate the particular language of a record in these tables. In our scenario, the records in L_COMMTEMPLATE and LONGDESCRIPTION are associated with LANGCODE ‘DE’, which indicates that the values are in German.

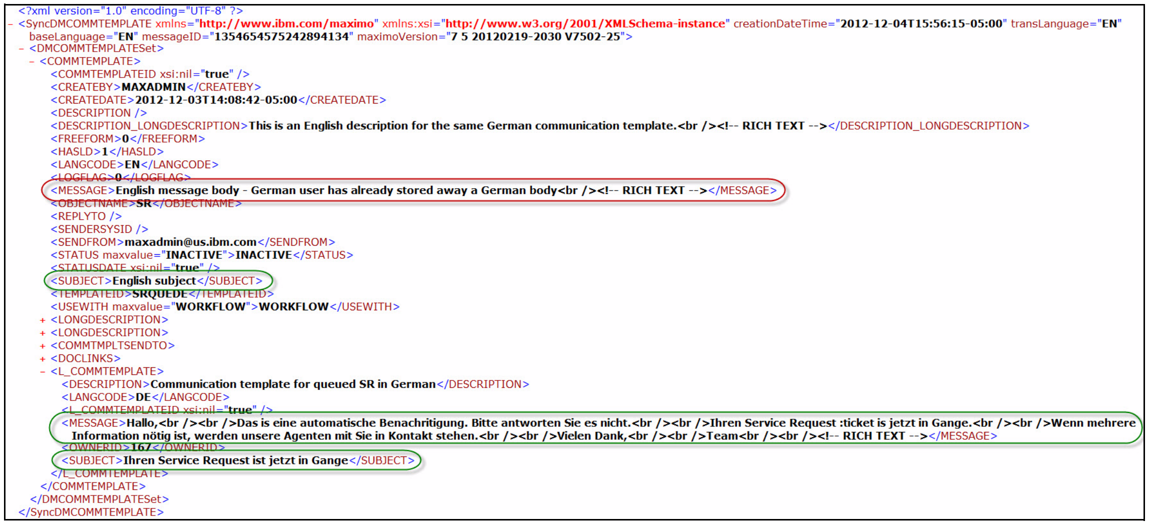
If the same three fields of the communication template also were associated with base language values, the English and German values are stored as shown in Table 10-4.

Table 15-4 Base and multi-language values storage for a communication template

Language-Field	**Database Table**	**Database Column**	**Language Code (LANGCODE column)**
EN - Subject	COMMTEMPLATE	SUBJECT	EN
DE- Subject	L_COMMTEMPLATE	SUBJECT	DE
EN - Message	COMMTEMPLATE	MESSAGE	EN
DE - Subject	L_COMMTEMPLATE	MESSAGE	DE
EN - Long description	LONGDESCRIPTION	LDTEXT	EN
DE - Long description	LONGDESCRIPTION	LDTEXT	DE

When a Migration Manager package is created to include this particular communication template, the English and German language values are automatically extracted and placed into the system configuration XML to ensure appropriate import processing during package deployment. Since the product environment’s base language is English, the base COMMTEMPLATE table’s columns hold the English values. Since German is the additional language, the L_COMMTEMPLATE table’s columns hold the German values. The LONGDESCRIPTION table is common to all languages and, thus, holds values of both English and German.

Figure 15-3 shows how the multi-language values are extracted in a Migration Manager package XML document.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SyncDMCOMMTEMPLATE xmlns="http://www.ibm.com/maximo" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" creationDateTime="2012-12-04T15:56:15-05:00" transLanguage="EN"
    baseLanguage="EN" messageID="1354654575242894134" maximoVersion="7 5 20120219-2030 V7502-25">
  - <DMCOMMTEMPLATESet>
    - <COMMTEMPLATE>
        <COMMTEMPLATEID xsi:nil="true" />
        <CREATEBY>MAXADMIN</CREATEBY>
        <CREATEDATE>2012-12-03T14:08:42-05:00</CREATEDATE>
        <DESCRIPTION />
        <DESCRIPTION_LONGDESCRIPTION>This is an English description for the same German communication template.<br /><!-- RICH TEXT --></DESCRIPTION_LONGDESCRIPTION>
        <FREEFORM>0</FREEFORM>
        <HASLD>1</HASLD>
        <LANGCODE>EN</LANGCODE>
        <LOGFLAG>0</LOGFLAG>
        <MESSAGE>English message body - German user has already stored away a German body<br /><!-- RICH TEXT --></MESSAGE>
        <OBJECTNAME>SR</OBJECTNAME>
        <REPLYTO />
        <SENDERSYSID />
        <SENDFROM>maxadmin@us.ibm.com</SENDFROM>
        <STATUS maxvalue="INACTIVE">INACTIVE</STATUS>
        <STATUSDATE xsi:nil="true" />
        <SUBJECT>English subject</SUBJECT>
        <TEMPLATEID>SRQUEDE</TEMPLATEID>
        <USEWITH maxvalue="WORKFLOW">WORKFLOW</USEWITH>
      + <LONGDESCRIPTION>
      + <LONGDESCRIPTION>
      + <COMMTMPLTSENDTO>
      + <DOCLINKS>
      - <L_COMMTEMPLATE>
          <DESCRIPTION>Communication template for queued SR in German</DESCRIPTION>
          <LANGCODE>DE</LANGCODE>
          <L_COMMTEMPLATEID xsi:nil="true" />
          <MESSAGE>Hallo,<br /><br />Das is eine automatische Benachritigung. Bitte antworten Sie es nicht.<br /><br />Ihren Service Request :ticket is jetzt in Gange.<br /><br />Wenn mehrere
            Information nötig ist, werden unsere Agenten mit Sie in Kontakt stehen.<br /><br />Vielen Dank,<br /><br />Team<br /><br /><!-- RICH TEXT --></MESSAGE>
          <OWNERID>167</OWNERID>
          <SUBJECT>Ihren Service Request ist jetzt in Gange</SUBJECT>
        </L_COMMTEMPLATE>
      </COMMTEMPLATE>
    </DMCOMMTEMPLATESet>
  </SyncDMCOMMTEMPLATE>
```

Figure 15-3 XML code of a communication template that holds German and English values

The deployment of a package with this communication template system configuration into a target environment results in multi-language data is stored in the same way as in the source environment. The German data values for this system configuration can be used in configuring and sending email notifications by an administrator who logs in to the target environment and choosing the German language.

Important: For a package that is created in the product environment with base language English and the added language German, a prerequisite for successful deployment into target requires that the target requirement also includes, at a minimum, base language English and the added language German. If the target does not include the added language German, deployment fails with the BMXAA4191E error.

Consider the following conditions before migration is performed:

- If more languages were enabled in the source environment, the information that is stored in the LANGUAGE table must be migrated to the target environment by using the DMLANGUAGE object structure before any other configurations are migrated. The DMLANGUAGE object structure is part of the DATADICTIONARY migration group. If a migration is performed from a source environment to a target environment that has fewer or other languages that are enabled, the migration fails.
- If a multi-language table was configured on site, determine whether there is a corresponding Migration Manager object structure that must be updated to include the multi-language table. The multi-language table must be added as a child of the base table for which it is configured.
- If the long description capability was configured for a business object, determine whether there is a corresponding Migration Manager object structure that must be updated to include the LONGDESCRIPTION table. The LONGDESCRIPTION table must be added as a child of the base table for which it is configured.

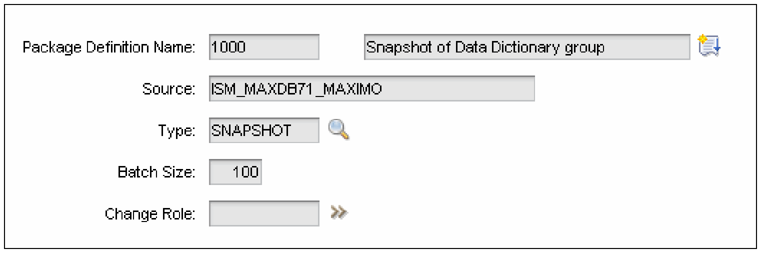
Important: When Migration Manager is used in multi-language environments, consider the following important points:

- If you modify an existing Migration Manager object structure to add a multi-language table in a source environment, you must modify the object structure of the target environment with the same change.
- Ensure that the unique ID column of the multi-language table is restricted in the Database Configuration application through the Restrict Attributes configuration. If column is not restricted and the Restrict Attributes configuration is disabled, restrict the unique ID column directly in the Object Structures application through the Inbound Setting Restrictions configuration.

15.3 Snapshot package versus Change package

There are two types of packages in the Migration Manager application: the Snapshot migration package and the Change migration package.

The Snapshot migration package contains configuration data content that exists in the source environment at the time that the package was created. Use the Snapshot package when there are significant changes in the environment. For example, use this package when you create an environment or create a set of applications and workflows. You can use the SQL WHERE clause to filter the content that you want to migrate in a Snapshot migration. Figure 15-4 shows an example of a Snapshot package definition.



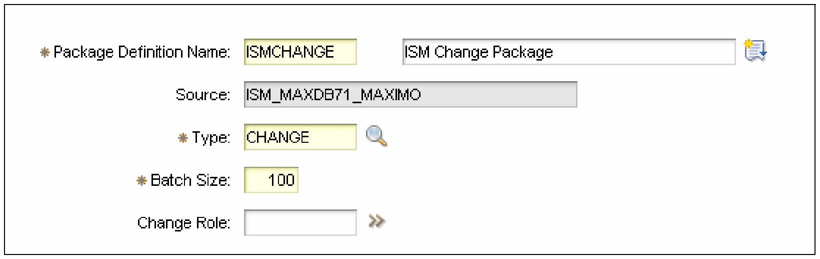
Package Definition Name:	1000	Snapshot of Data Dictionary group
Source:	ISM_MAXDB71_MAXIMO	
Type:	SNAPSHOT	
Batch Size:	100	
Change Role:		

Figure 15-4 Definition of a Snapshot package

Tip: Use the 1=0 SQL statement to migrate no content in the selected migration group SQL.

To use Snapshot packages, the entire migration requirement must be clearly defined, identified, and documented to the exact name of the content elements. The configuration of Snapshot packages requires a considerable amount of configuration on the source environment.

Change migration packages often contain less content data in comparison to the Snapshot packages. Change packages contain the configuration data that was created since the package was activated. Figure 15-5 shows an example of a Change package definition.



* Package Definition Name:	ISMCHANGE	ISM Change Package
Source:	ISM_MAXDB71_MAXIMO	
* Type:	CHANGE	
* Batch Size:	100	
Change Role:		

Figure 15-5 Definition of a Change package

Type: After you define the type field, it becomes read-only and cannot be changed.

In Change package definitions, the Change Role field can be edited. By using this field, you can track the changes of a specific role.

Change packages are useful when there already is a source environment and a target environment that are in synch at the time that this type of package is activated. Separate Change packages must not overlap. For example, separate Change packages must not listen to the same objects. It is useful if each developer has its own Change package.

Change packages: Change packages cannot track changes when Admin mode is turned on. Admin mode shuts off all of the event broadcasts. If a Change package is active, you receive a warning when you turn on Admin mode.

Certain object structures that are created to be used with a Snapshot package cannot be used for Change packages. The Migration Manager cannot handle certain data in a Change package. For example, hierarchical data cannot be migrated with a Change package. The Change package does not know the hierarchical order, which causes validation errors. Classifications, locations, and assets are examples of hierarchical data. Overlapped packages, which have the same content, can cause an error for the same reason. Always preview the deployment changes before deployment to see whether the content is valid.

DELETE statements: Snapshot packages cannot track DELETE statements because they do not keep historical data. If you want to keep track of deleted configuration content, use a Change package instead of a Snapshot package.

Complete the following steps to see and manage the changes in a Change package:

1. Browse to the Migration Manager application by selecting **Go To** → **System Configuration** → **Migration** → **Migration Manager**.
2. Select the related Change package, and then select **View Event Tracking Records** in the Select Action menu, as shown in Figure 15-6.

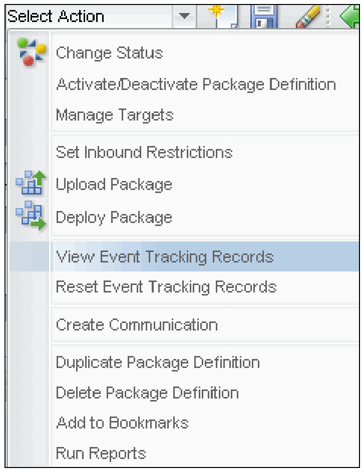
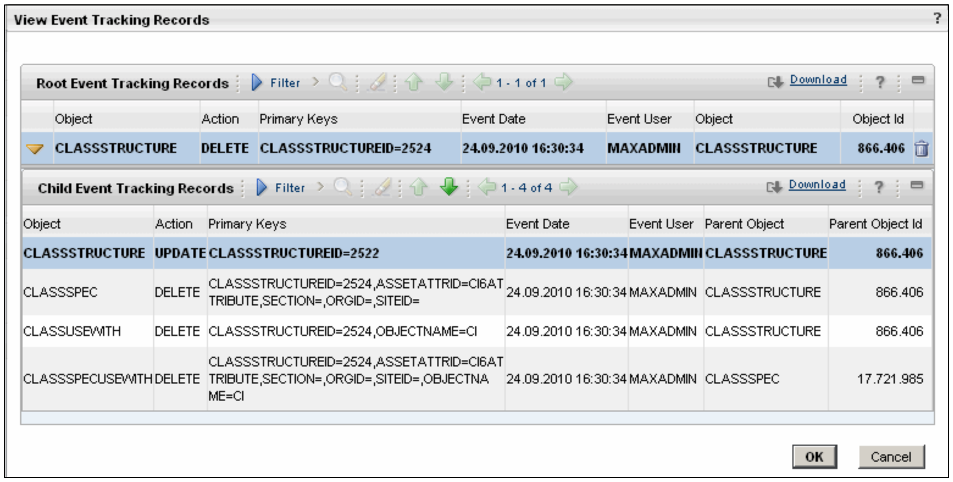

Select Action
Change Status
Activate/Deactivate Package Definition
Manage Targets
Set Inbound Restrictions
Upload Package
Deploy Package
View Event Tracking Records
Reset Event Tracking Records
Create Communication
Duplicate Package Definition
Delete Package Definition
Add to Bookmarks
Run Reports

Figure 15-6 View Event Tracking Records option

3. Figure 15-7 shows the event tracking records for a Change package. The figure shows which objects and related objects are altered. If there are certain records that you do not want to migrate, click the Trash Can icon to the right of the line. Child event tracking records are deleted automatically.


View Event Tracking Records
Root Event Tracking Records
Filter
1 - 1 of 1
Download
Child Event Tracking Records
Filter
1 - 4 of 4
Download
OK
Cancel

Object	Action	Primary Keys	Event Date	Event User	Object	Object Id
CLASSSTRUCTURE	DELETE	CLASSSTRUCTUREID=2524	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406

Object	Action	Primary Keys	Event Date	Event User	Parent Object	Parent Object Id
CLASSSTRUCTURE	UPDATE	CLASSSTRUCTUREID=2522	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406
CLASSSPEC	DELETE	CLASSSTRUCTUREID=2524,ASSETATTRID=CI6ATTRIBUTE,SECTION=,ORGID=,SITEID=	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406
CLASSUSEWITH	DELETE	CLASSSTRUCTUREID=2524,OBJECTNAME=CI	24.09.2010 16:30:34	MAXADMIN	CLASSSTRUCTURE	866.406
CLASSSPECUSEWITH	DELETE	CLASSSTRUCTUREID=2524,ASSETATTRID=CI6ATTRIBUTE,SECTION=,ORGID=,SITEID=,OBJECTNAME=CI	24.09.2010 16:30:34	MAXADMIN	CLASSSPEC	17.721.985

Figure 15-7 Events that are recorded in a Change package

4. If you want to delete all of the event tracking records, select **Reset Event Tracking Records** in the Select Action menu, as shown in Figure 15-8.

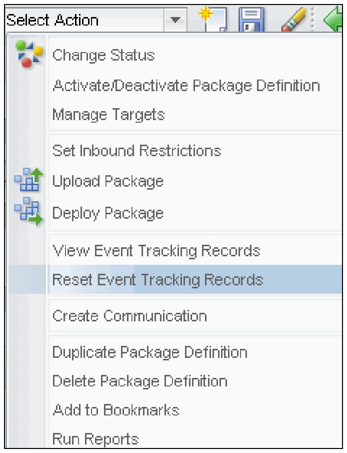
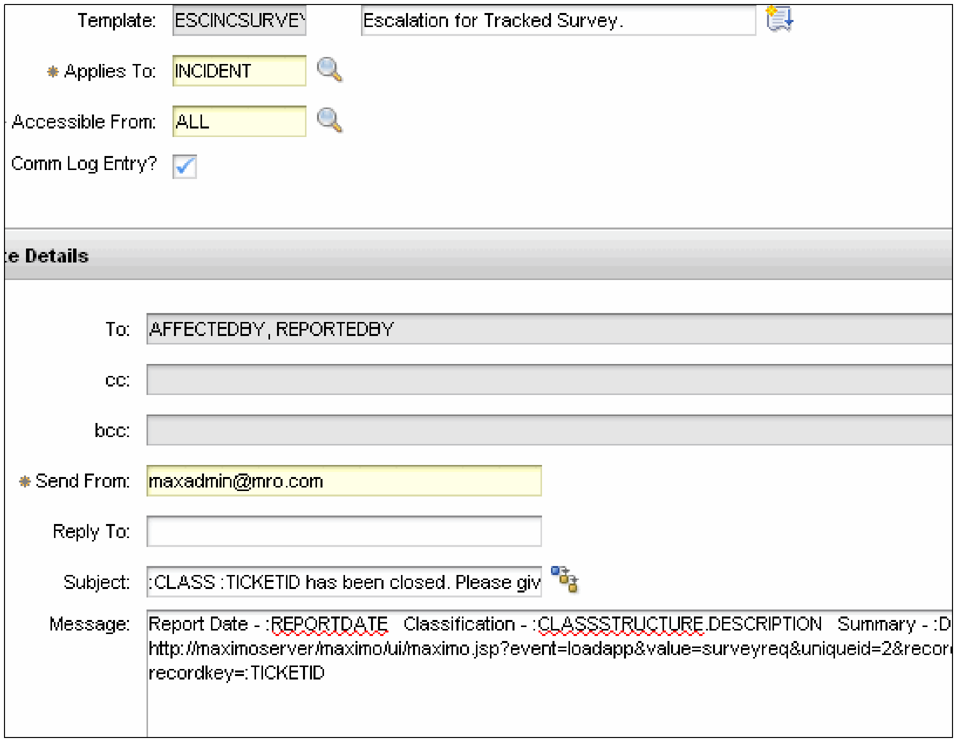

Select Action
Change Status
Activate/Deactivate Package Definition
Manage Targets
Set Inbound Restrictions
Upload Package
Deploy Package
View Event Tracking Records
Reset Event Tracking Records
Create Communication
Duplicate Package Definition
Delete Package Definition
Add to Bookmarks
Run Reports

Figure 15-8 Reset Event Tracking Records option

5. When you select this option, a confirmation message is displayed. Click **Yes** if you are sure that you want to delete the tracked events.

15.4 Embedded URLs

Certain configuration data can include embedded URLs. For example, you want to send users an email with a link to the system. You can use the Communication Templates application. Figure 15-9 shows an example of a communication template with an embedded URL.



Template:	ESCINCSURVEY Escalation for Tracked Survey.
* Applies To:	INCIDENT
Accessible From:	ALL
Comm Log Entry?	✓

e Details

To:	AFFECTEDBY, REPORTEDBY
cc:	
bcc:	
* Send From:	maxadmin@mro.com
Reply To:	
Subject:	:CLASS :TICKETID has been closed. Please giv
Message:	Report Date - :REPORTDATE Classification - :CLASSSTRUCTURE.DESCRIPTION Summary - :D http://maximoserver/maximo/ui/maximo.jsp?event=loadapp&value=surveyreq&uniqueid=2&recor recordkey=:TICKETID

Figure 15-9 Example of a communication template with an embedded URL

URLs that are used in communication templates, endpoints, and launch-in-contexts generally differ in separate environments.

You must correct these URLs manually after these types of objects are migrated. Communication templates are INACTIVE after migration. Check the URLs in the communication templates, and then change their status to ACTIVE. Other objects, such as launch-in-context and endpoints, are stateless. Be careful before you use these objects in the target environment.

15.5 Clustered environment considerations

There are many scenarios in which clustered environments behave differently.

Packages can contain structural and non-structural content. Structural content and part of the non-structural content are cached by the Java virtual machine (JVM) to provide performance. The following examples are types of cached content:

- Table, view, and column structures
- Application presentations
- Domains

In a single-server environment, you can migrate these types of content. You can use this content when the migration completes.

In a clustered environment, each member JVM server connects to the same back-end database to read and load this content into its cache. When a new package is deployed, cached content must be refreshed on each clustered server.

Restart each JVM in the cluster to ensure that the changes in the migration package are reflected on each JVM. Otherwise, the users that are using the other JVMs are not affected by the changes. In clustered environments, changes must be scheduled (by using an outage window) to restart all JVMs. If a full outage window cannot be scheduled, you can use a *partial* or *rolling* outage approach in which JVM servers are restarted in phases, and one or more servers always are available. This approach requires the use of a load balancer appliance.

Tip: During the deployment of packages in full outage windows, stop all of the JVMs in the cluster, except the deployment JVM that is used through the end of the entire migration process. After you complete the migration process, be sure that the Admin mode is off and that the JVMs are started.

Warning: The first users that log in to each JVM experience slow response time until the server builds its cache. After the cache is built, the performance improves.

15.6 Change tracking and ad hoc reporting

A common misconception is that the Migration Manager is considered an afterthought and that it is a tool to use only after configuration is complete.

In most scenarios, a Snapshot package is used to promote configuration changes. However, the developer who performs the migration is not always the original developer, or it might be a multiple developer project. As we see in the examples that are described in this publication, the migration requirements must be clearly defined, identified, and documented to the exact names of the configuration elements to use the Snapshot package effectively.

This section describes how the Change package in the Migration Manager is an important element in the development cycle and how it can be used effectively.

15.6.1 When to use Change packages

Change packages are useful when the source environment and target environment are synchronized and the development efforts are not started. However, to use the Change package effectively, the Migration Manager must be part of the development planning to harness its capabilities during the development cycle.

The simplest and ideal scenario is when the development planning determines that Change packages capture the configuration activities in a development environment and then deploy them to a target environment on an organized way.

Roles are assigned and package content definitions are created. Most importantly, the developers' roles do not overlap, which means that no two developers work on the same configuration content, even if separate Change packages are listening to the same migration objects.

If roles are not necessary (for example, there is only one developer), it is better to define only one Change Package that contains all of the migration objects that were identified as part of the migration requirements. Deploying this type of package performs all of the tracked configuration events in one deployment.

You also can define several packages that each track only one specific migration group, without overlapping the migration objects. The deployment of Change packages of this type requires taking into account the dependency of groups in the sequence of the deployment. It might also require coordinated efforts when partially completed development must be promoted to create the package. Then, you reset the package and activate the package to continue development.

15.6.2 Using Change packages to track development activities

The development environment is a shared environment with multiple developers who are working in a coordinated manner to configure the product to meet product solution requirements in a specific time frame.

In this scenario, development activities often begin rapidly. The planning and incorporation of the Migration Manager Change packages often do not happen at the beginning of Migration Manager implementations.

In most Migration Manager implementations, the Snapshot migration package functionality is used to promote configuration. The use of SQL WHERE clause statements to target specific configuration content is critical for the promotion of specific content. The challenge is to define this content and identify the names of configuration key values.

The records that must be migrated identified by using a naming convention (for example, object key values that begin with prefix CUST) or be clearly documented in the form of spreadsheets or other tabular-type documents. Without this information, it takes considerable time and effort to proceed with migration.

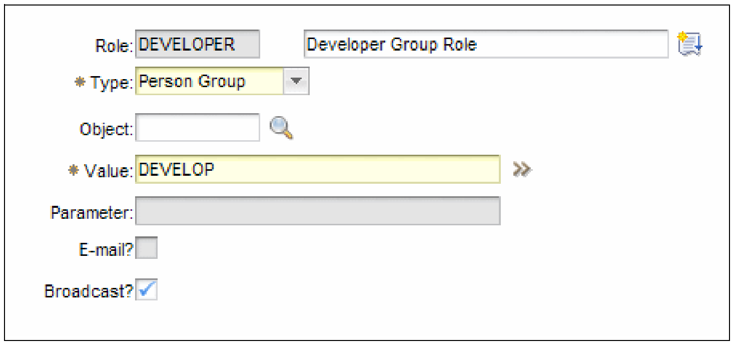
By using the Change package, you can capture database events (add, update, or delete) those effect-specific configuration content records. This functionality can be used as a tracking device only, without creating, distributing, and deploying. Then, a Snapshot package can be created that is based on the changes that are tracked.

15.6.3 Configuring change roles

Change package functionality tracks changes that are made by specific users by using *roles*, which is useful in multiple developer teams. Each developer is assigned a user ID to log in and perform a configuration.

A role can be assigned to a person or to a person group. It is up to the team to decide the type of group to use.

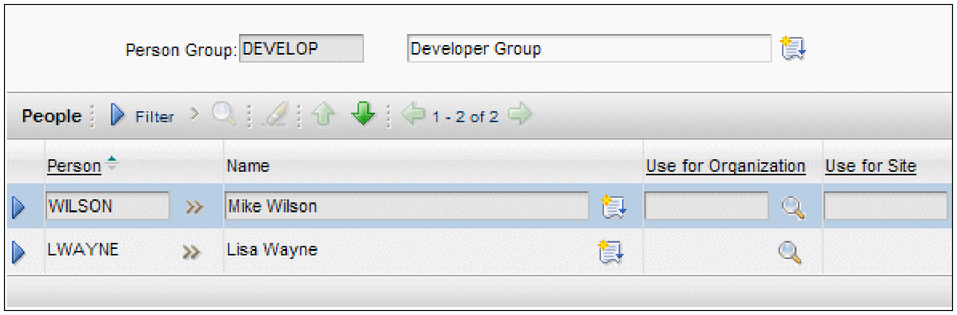
For this example, we create a DEVELOPER role of type Person Group, as shown in Example 15-10, which is based on the Person Group that is called DEVELOP, as shown is shown in Example 15-11. There are two person IDs assigned to this group, and these IDs correspond to the user IDs of the two developers. In our example, the users are WILSON and LWAYNE.



Role:	DEVELOPER — Developer Group Role
* Type:	Person Group
Object:	
* Value:	DEVELOP
Parameter:	
E-mail?	
Broadcast?	✓

Figure 15-10   DEVELOPER role

Important: In Figure 15-10, if the broadcast flag is left cleared, the Migration Manager does not record the events that are triggered by all members of the group. Only the events of the designated default person of the group are recorded.



Person Group: DEVELOP — Developer Group

Person	Name	Use for Organization	Use for Site
WILSON	Mike Wilson		
LWAYNE	Lisa Wayne		

Figure 15-11   DEVELOP person group

You also can use person type roles in Change packages. The advantage of a person group is the ability to add or remove users in a dynamic environment.

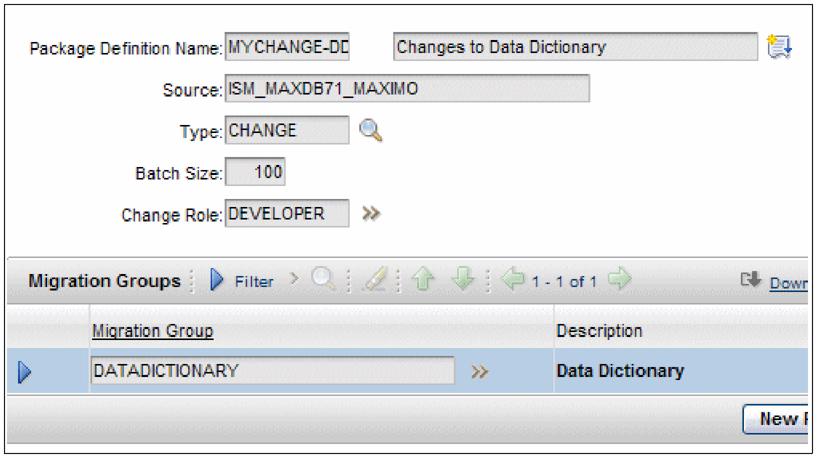
The two roles that are defined here are for two developers who configure changes in applications that affect the data dictionary and the application framework.

In this example, their work does not overlap. WILSON changes the objects in the data dictionary group and LWAYNE makes the configuration changes on the application side. This scenario is an example of how you can organize the development. Both developers can perform configurations in both groups. The important point here is the ability to track these events.

15.6.4 Change package definitions

After the roles are defined, it is time to create the Change packages. These packages track the changes to specific configuration objects that were made by the users that are defined in the role that is assigned to the package.

In this example, a Change package definition is created with the standard DATADICTIONARY migration group to track changes to any object structure that is included in this group, as shown in Figure 15-12. This definition is associated with the DEVELOPER role.

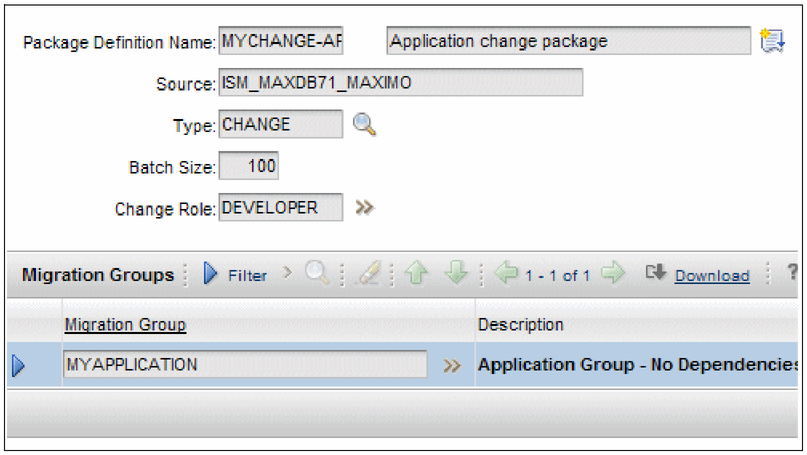


Package Definition Name:	MYCHANGE-DD — Changes to Data Dictionary
Source:	ISM_MAXDB71_MAXIMO
Type:	CHANGE
Batch Size:	100
Change Role:	DEVELOPER

Migration Group	Description
DATADICTIONARY	Data Dictionary

Figure 15-12 Change package MYCHANGE-DD

A second Change package definition is created to track changes to applications. A new migration group is used as a duplicate copy of the APPLICATIONS migration group. This new group is called MYAPPLICATION, and all dependent groups are removed. This package also is associated with the DEVELOPER role, as shown in Figure 15-13.


Package Definition Name: MYCHANGE-AF Application change package

Source: ISM_MAXDB71_MAXIMO

Type: CHANGE

Batch Size: 100

Change Role: DEVELOPER

Migration Group	Description
MYAPPLICATION	Application Group - No Dependencie

Figure 15-13 Change package MYCHANGE-APP

Tracking changes: Change tracking is performed for every Maximo business object (MBO) for the migration group that is specified in the list. More groups can be added, as needed, if the package definition status is WAPPR.

15.6.5 Tracking change events

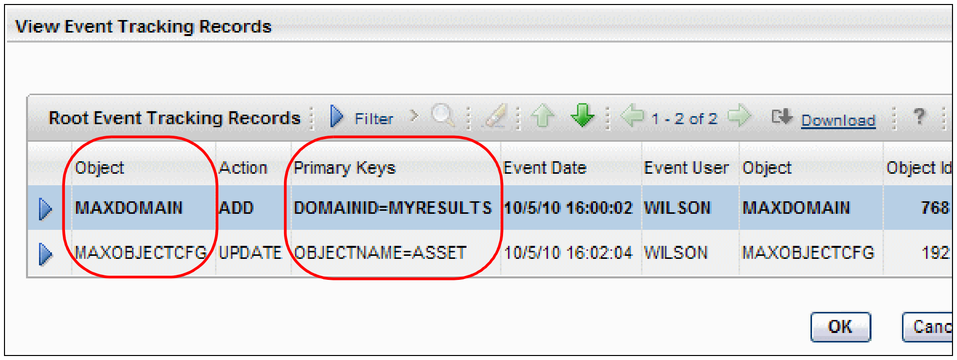
After the packages are approved and activated, the developers can start the configuration efforts. These events are tracked and stored in the event tables.

When structural changes are made to the database by using the Database Configuration application, applying these changes requires the Admin mode to be turned on. During this time, the tracking stops. Do not perform any changes until the database configuration is complete and Admin mode is turned off in the system.

We demonstrate how we use the Change package for tracking. The following changes are required:

- A new domain, MYRESULTS, of type ALN was added. The values are PASSED and FAILED.
- A new attribute that is called MYRESULTS was created in the ASSET table. This attribute holds the results of test values. The new domain is configured for validation.
- The new attribute for test results is added to the ASSET application and configured to validate.

After the developers complete their configuration, we can now see the exact events from the Change package definitions by clicking **Select Action** → **View Event Tracking Records** (as shown in Figure 15-14 and Figure 15-15).

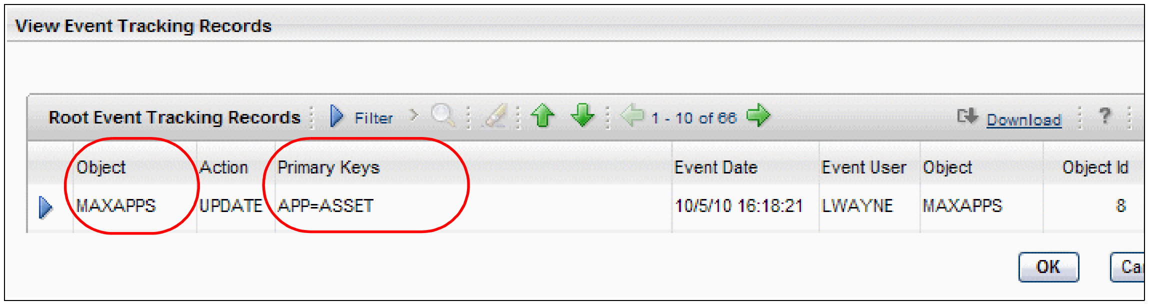

View Event Tracking Records

Root Event Tracking Records | Filter | 1 - 2 of 2 | Download

Object	Action	Primary Keys	Event Date	Event User	Object	Object Id
MAXDOMAIN	ADD	DOMAINID=MYRESULTS	10/5/10 16:00:02	WILSON	MAXDOMAIN	768
MAXOBJECTCFG	UPDATE	OBJECTNAME=ASSET	10/5/10 16:02:04	WILSON	MAXOBJECTCFG	192

OK | Canc

Figure 15-14   Events for package MYCHANGE-DD


View Event Tracking Records

Root Event Tracking Records | Filter | 1 - 10 of 66 | Download

Object	Action	Primary Keys	Event Date	Event User	Object	Object Id
MAXAPPS	UPDATE	APP=ASSET	10/5/10 16:18:21	LWAYNE	MAXAPPS	8

OK | Ca

Figure 15-15   Events for package MYCHANGE-APP

This example shows the events' sample configurations that were created by the two developers. Each package definition provides the functionality to list these recorded events in the application.

Single role Change package: Depending on the strategy, you can use a single role Change package to track change events. The single Change package must include all of the migration groups that were identified as configured.

The Object and the Primary Keys columns are the most important columns for tracking change events. With this information, we can define the migration requirements for the Snapshot package.

For example, from the events that are shown in Figure 15-14 on page 374, we need the objects and SQL conditions that are shown in Table 15-5 for our package.

Table 15-5 Where clauses for data dictionary objects in a Snapshot package

MIgration object	Object	Where clause
DMMAXDOMAIN	MAXDOMAIN	domainid in ('MYRESULTS')
DMMAXOBJECTCFG	MAXOBJECTCFG	objectname in ('ASSET')

From the events that are shown in Figure 15-15 on page 374, we can gather enough information to determine the objects and SQL conditions (as shown in Table 15-6) that are needed for a Snapshot package.

Table 15-6 Where clauses for application objects in a Snapshot package

MIgration object	Object	Where clause
DMMAXAPPS	MAXAPPS	`app in ('ASSET')`

In a multiple developer environment, with several configurations that take place at the same time, developers often must learn other tools or applications and focus on completing tasks. The use of Change package functionality provides essential information (which sometimes is not documented) to prepare and deploy Snapshot packages in an ongoing basis throughout the development cycle, or as part of the daily operations and maintenance tasks.

Tracking what is changed and by whom is a good practice and enables accountability and change process documentation. However, this tracking information is stored in the database, and the only documentation might be in the form of a tracking spreadsheet.

Next, we show how you can extract this information for documentation and Snapshot package preparation purposes.

15.6.6 Creating ad hoc reports of change events

You can configure ad hoc or query-based report (QBR) to gather all of the changes that are tracked for all users across all packages. You must perform several one-time tasks to configure and run this type of ad hoc report.

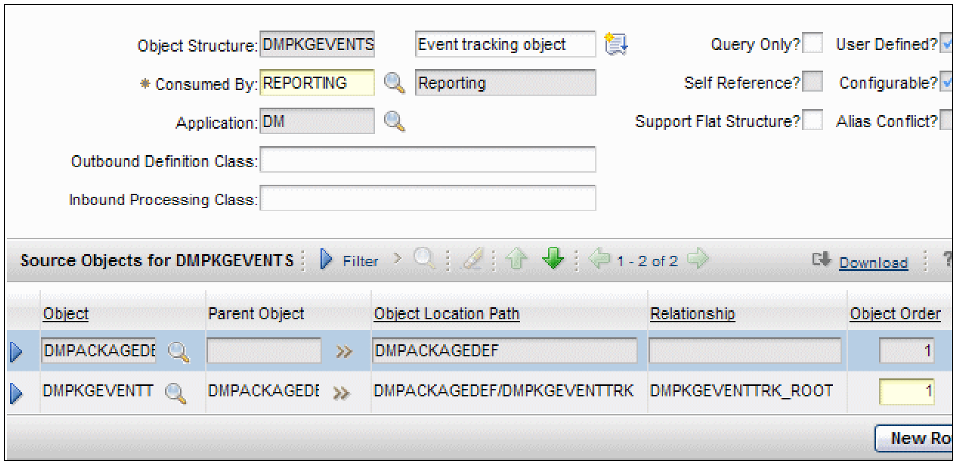
Defining reporting object structure

Create a DMPKGEVENTS object structure by clicking **Go To** → **System Configuration** → **Migration** → **Object Structures**.

Complete the following steps to apply details to the new object structure:

1. Specify the Consumed By field for the DMPKGEVENTS object structure as REPORTING.
2. Specify the DMPACKAGEDEF MBO as the parent and DMPKGEVENTTRK MBO as the child.
3. For the DMPKGEVENTTRK child MBO, specify the Relationship as DMPKGEVENTTRK_ROOT. This specification results in the ad hoc report displaying only the changes to root (or primary) MBOs.
4. Save the object structure.

The new object structure is shown in Figure 15-16.



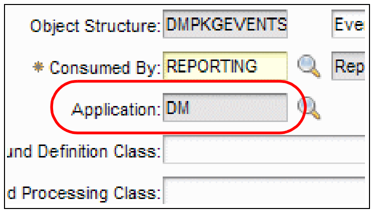
Object Structure:	DMPKGEVENTS	Event tracking object	Query Only? ☐	User Defined? ✓
* Consumed By:	REPORTING	Reporting	Self Reference? ☐	Configurable? ✓
Application:	DM		Support Flat Structure? ☐	Alias Conflict? ☐
Outbound Definition Class:				
Inbound Processing Class:				

Source Objects for DMPKGEVENTS

Object	Parent Object	Object Location Path	Relationship	Object Order
DMPACKAGEDE		DMPACKAGEDEF		1
DMPKGEVENTT	DMPACKAGEDE	DMPACKAGEDEF/DMPKGEVENTTRK	DMPKGEVENTTRK_ROOT	1

Figure 15-16 DMPKGEVENTS object structure for reporting

This ability to report object association to an application is new in Maximo Base Services Fix Pack 7.1.1.6, as shown in Figure 15-17. In Fix Pack 7.1.1.5, this association is done in Application Designer.



Object Structure:	DMPKGEVENTS	Eve
* Consumed By:	REPORTING	Rep
Application:	DM	
ınd Definition Class:		
d Processing Class:		

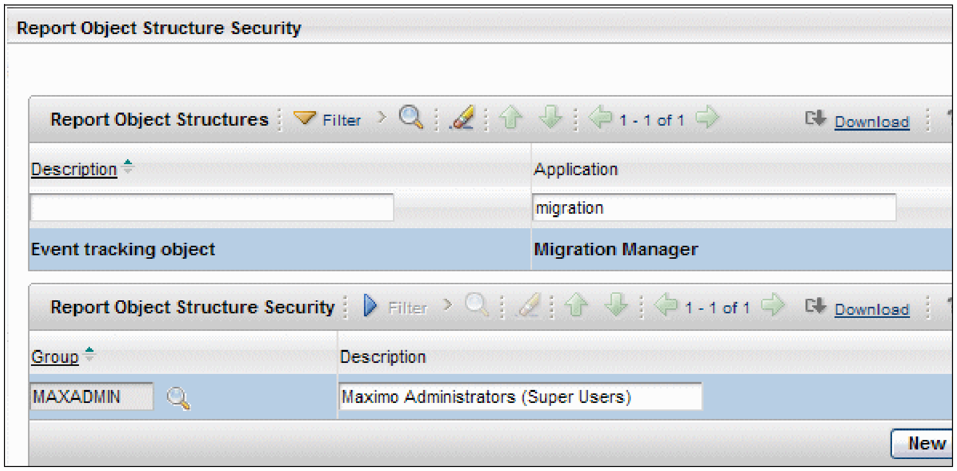
Figure 15-17 Application for reporting object structure

After you complete this step, a new reporting icon is added to the Migration Manager application.

Granting security access to reporting object structure

Complete the following steps to grant security access to users of Maximo Base Services Fix Pack 7.1.1.6:

1. Click **Go To** → **Administration** → **Reporting** → **Report Administration**. From the List tab, click **Select Action** → **Set Report Object Structure Security**, and filter the application Migration Manager.
2. Add the `MAXADMIN` group to the security list and click **OK**. The security object is shown in Figure 15-18.



Report Object Structure Security

Report Object Structures | Filter | 1 - 1 of 1 | Download

Description	Application
	migration
Event tracking object	Migration Manager

Report Object Structure Security | Filter | 1 - 1 of 1 | Download

Group	Description
MAXADMIN	Maximo Administrators (Super Users)

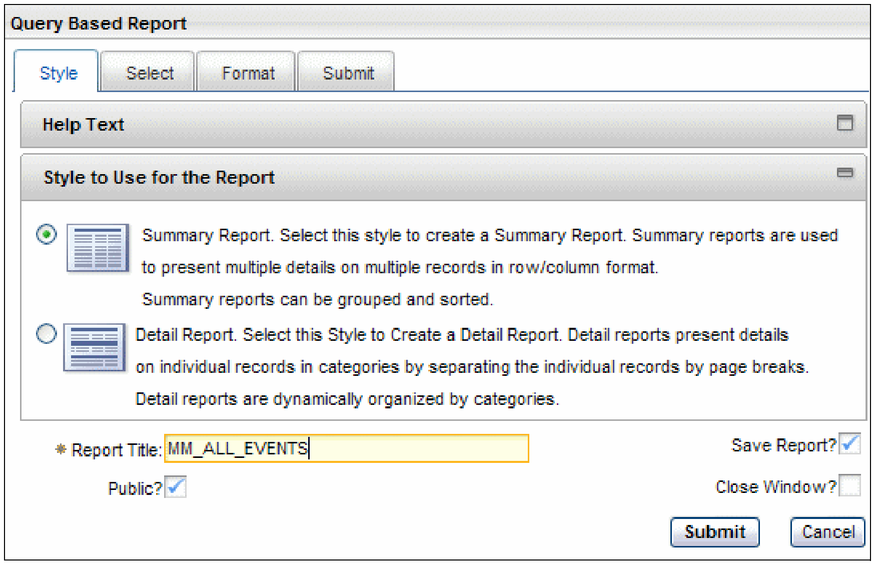
Figure 15-18 Report Object Structure Security window

Configuring the Ad Hoc report

In the Migration Manager application, click the **Create Report** icon to display the Query-based report pop-up window. In Maximo Base Services Fix Pack 7.1.1.6, the window features multiple tabs.

Complete the following steps to configure the Style tab, as shown in Figure 15-19:

1. Specify a name for the ad hoc report (`MM_ALL_EVENTS`).
2. Make it public by selecting the **Public?** option.
3. Select **Save Report** to enable the report design to be available to users and sessions.
4. Select **Summary Report** to display records in a simple row/column format.



[✓] Summary Report. Select this style to create a Summary Report. Summary reports are used to present multiple details on multiple records in row/column format. Summary reports can be grouped and sorted.

[ ] Detail Report. Select this Style to Create a Detail Report. Detail reports present details on individual records in categories by separating the individual records by page breaks. Detail reports are dynamically organized by categories.

* Report Title: MM_ALL_EVENTS

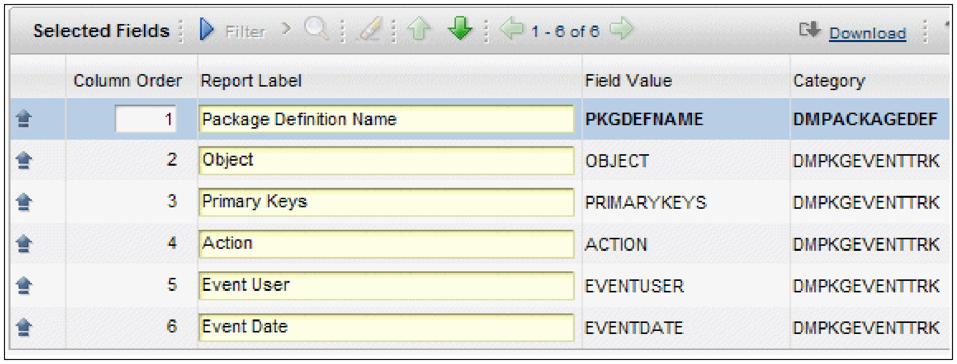
Public? ✓

Save Report? ✓

Close Window? ☐

Figure 15-19 MM_ALL_EVENT query-based report Style tab

5. On the Select tab, configure the selected fields list, as shown in Figure 15-20.

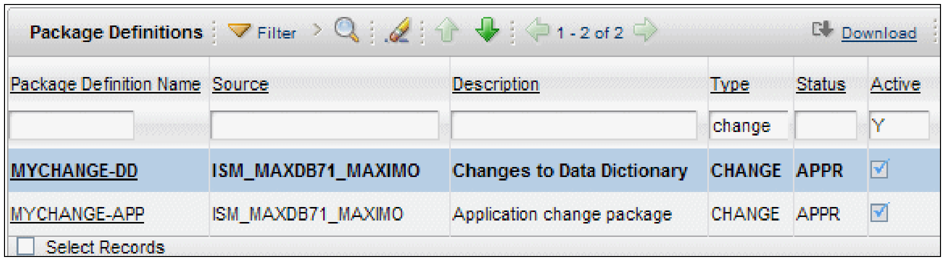

Selected Fields | Filter | 1 - 6 of 6 | Download

	Column Order	Report Label	Field Value	Category
	1	Package Definition Name	PKGDEFNAME	DMPACKAGEDEF
	2	Object	OBJECT	DMPKGEVENTTRK
	3	Primary Keys	PRIMARYKEYS	DMPKGEVENTTRK
	4	Action	ACTION	DMPKGEVENTTRK
	5	Event User	EVENTUSER	DMPKGEVENTTRK
	6	Event Date	EVENTDATE	DMPKGEVENTTRK

Figure 15-20 *MM_ALL_EVENTS selected fields for QBR*

6. By leaving the option **Apply the Current Query and Filter from the Application?** selected, you can use the filtering of package definitions from the Migration Manager application List tab.

In our example, the filtering was done on the package Type and Active fields, as shown in Figure 15-21.


Package Definitions | Filter | 1 - 2 of 2 | Download

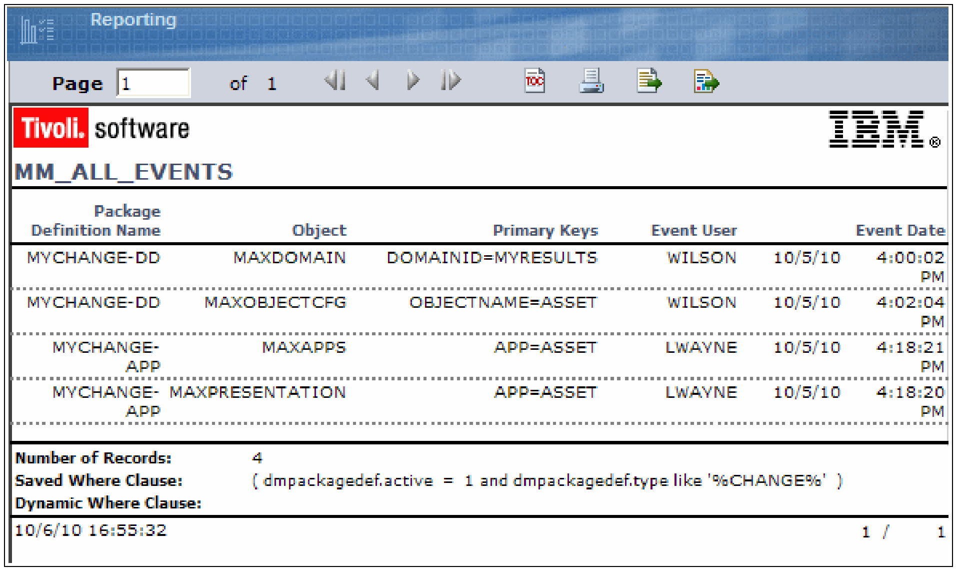
Package Definition Name	Source	Description	Type	Status	Active
			change		Y
MYCHANGE-DD	ISM_MAXDB71_MAXIMO	Changes to Data Dictionary	CHANGE	APPR	✓
MYCHANGE-APP	ISM_MAXDB71_MAXIMO	Application change package	CHANGE	APPR	✓

☐ Select Records

Figure 15-21 *Change Package list tracking events*

Running the ad hoc report

You can now submit the report to run immediately as a PDF report. After you close the definition, you can start the report in the Migration Manager application by clicking **Select Action** → **Run Reports** and then clicking the saved report **MM_ALL_EVENTS**. Figure 15-22 shows the report output.



MM_ALL_EVENTS

Package Definition Name	Object	Primary Keys	Event User	Event Date	
MYCHANGE-DD	MAXDOMAIN	DOMAINID=MYRESULTS	WILSON	10/5/10	4:00:02 PM
MYCHANGE-DD	MAXOBJECTCFG	OBJECTNAME=ASSET	WILSON	10/5/10	4:02:04 PM
MYCHANGE-APP	MAXAPPS	APP=ASSET	LWAYNE	10/5/10	4:18:21 PM
MYCHANGE-APP	MAXPRESENTATION	APP=ASSET	LWAYNE	10/5/10	4:18:20 PM

Number of Records: 4
Saved Where Clause: (dmpackagedef.active = 1 and dmpackagedef.type like '%CHANGE%')
Dynamic Where Clause:

10/6/10 16:55:32 1 / 1

Figure 15-22 MM_ALL_EVENTS query-based report output

The report can be scheduled for running and emailed to recipients. The contents of the report can be output as a spreadsheet (XLS) or an Adobe PDF document.

15.6.7 Change tracking versus migration collection tracking

With the introduction of the Migration Collection features, a second tracking approach is available to implementers. The approaches are similar in the following respects:

- Both approaches offer a listener-based tracking capability.
- Both approaches offer the implementer the ability to stop and start the tracking as needed.
- Both approaches offer the implementer the ability to review tracked data and remove specific tracking data before a package is created.

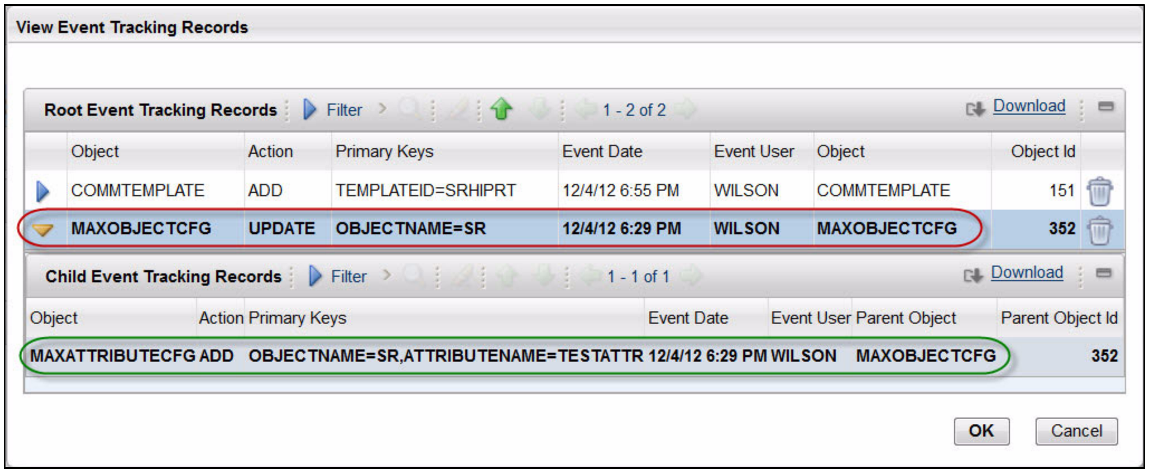
There are key differences between the two tracking approaches and the resulting XML format at the time the package is created, as shown in Table 15-7.

Table 15-7 Key differences between change and migration collection tracking

Change Tracking	**Migration Collection tracking**
Pre-requisites for change tracking: ► Package definition of type CHANGE ► Requisite migration groups included in the definition ► Package definition that is approved and activated	Pre-requisites for migration collection tracking: ► Migration collection definition ► Application and corresponding object structure chosen ► Database events to track chosen
Tracking records are stored in the DMEVENTTRK table; each record includes the specific database event that affects the record.	No separate tracking records stored. Records are added or removed directly from the collection in response to the corresponding database event (in the DMCOLLECTIONOBJ table).
Tracking is more granular; tracking records are created for main object events and child object events.	Tracking is less granular; only events on main objects are tracked.
Package specifies more granular action tags; package creation causes event type (add, update, or delete) to be specified automatically as actions for the corresponding XML-formatted data are extracted for a business object.	Package specifies less granular action tags; action is manually specified as AddChange or Replace for the entire package definition, which is similar to a snapshot package definition.

These differences highlight the fact that the goal with Migration Collection approach is data collection in the source environment, whereas the goal with Change tracking is to capture incremental changes and propagate those changes to the target environment. Understanding these differences enables the implementer to use the correct tools to capture and package development environment system configurations.

Figure 15-23 shows granular change tracking records that represent a new attribute that was added to an existing business object. The tracking record for the SR business object is associated with the UPDATE action, while the tracking record for the TESTATTR attribute is associated with the ADD action.


View Event Tracking Records

Root Event Tracking Records | Filter | 1 - 2 of 2 | Download

Object	Action	Primary Keys	Event Date	Event User	Object	Object Id
COMMTEMPLATE	ADD	TEMPLATEID=SRHIPRT	12/4/12 6:55 PM	WILSON	COMMTEMPLATE	151
MAXOBJECTCFG	**UPDATE**	**OBJECTNAME=SR**	**12/4/12 6:29 PM**	**WILSON**	**MAXOBJECTCFG**	**352**

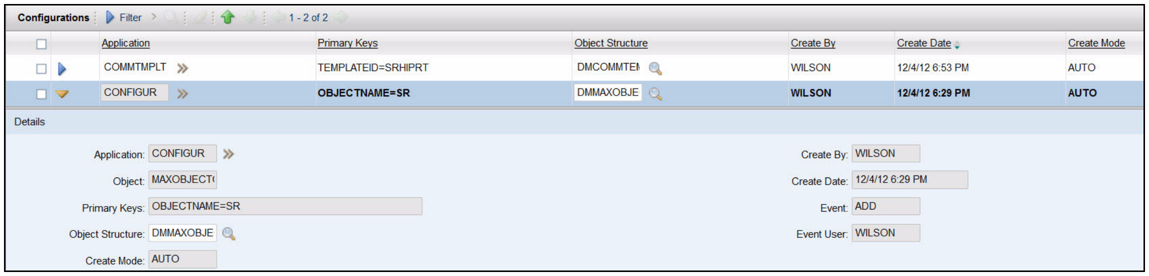
Child Event Tracking Records | Filter | 1 - 1 of 1 | Download

Object	Action	Primary Keys	Event Date	Event User	Parent Object	Parent Object Id
MAXATTRIBUTECFG	**ADD**	**OBJECTNAME=SR,ATTRIBUTENAME=TESTATTR**	**12/4/12 6:29 PM**	**WILSON**	**MAXOBJECTCFG**	**352**

OK | Cancel

Figure 15-23   Change tracking records for main object and child object

Figure 15-24 shows the less granular migration collection tracking records that are nothing more than collection records for the main object. There are no records that pertain to child object events.


Configurations | Filter | 1 - 2 of 2

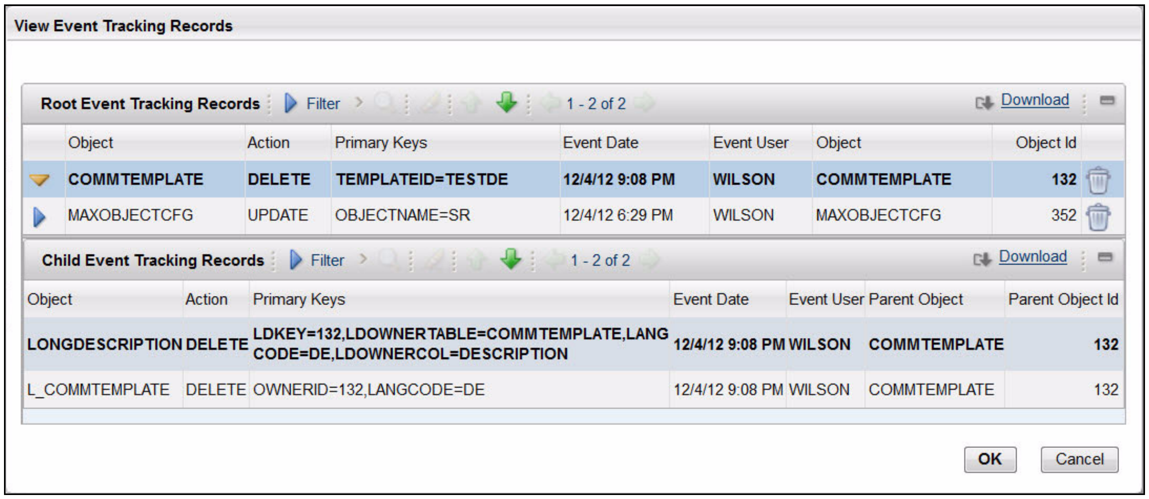
Application	Primary Keys	Object Structure	Create By	Create Date	Create Mode
COMMTMPLT	TEMPLATEID=SRHIPRT	DMCOMMTEI	WILSON	12/4/12 6:53 PM	AUTO
CONFIGUR	**OBJECTNAME=SR**	DMMAXOBJE	**WILSON**	**12/4/12 6:29 PM**	**AUTO**

Details

Application:	CONFIGUR	Create By:	WILSON
Object:	MAXOBJECT(	Create Date:	12/4/12 6:29 PM
Primary Keys:	OBJECTNAME=SR	Event:	ADD
Object Structure:	DMMAXOBJE	Event User:	WILSON
Create Mode:	AUTO		

Figure 15-24   Collection tracking records for main object

Figure 15-25 shows how change tracking captures delete events for an object. The TESTDE communication template is deleted, which results in the delete of a long description record and a multi-language record in German (LANGCODE=DE), which are both associated with the communication template.



View Event Tracking Records

Root Event Tracking Records | Filter | 1 - 2 of 2 | Download

Object	Action	Primary Keys	Event Date	Event User	Object	Object Id
COMMTEMPLATE	**DELETE**	**TEMPLATEID=TESTDE**	**12/4/12 9:08 PM**	**WILSON**	**COMMTEMPLATE**	**132**
MAXOBJECTCFG	UPDATE	OBJECTNAME=SR	12/4/12 6:29 PM	WILSON	MAXOBJECTCFG	352

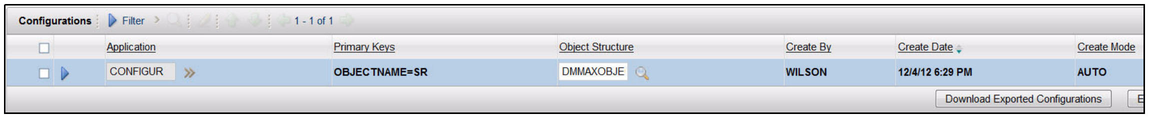
Child Event Tracking Records | Filter | 1 - 2 of 2 | Download

Object	Action	Primary Keys	Event Date	Event User	Parent Object	Parent Object Id
LONGDESCRIPTION	**DELETE**	**LDKEY=132,LDOWNERTABLE=COMMTEMPLATE,LANGCODE=DE,LDOWNERCOL=DESCRIPTION**	**12/4/12 9:08 PM**	**WILSON**	**COMMTEMPLATE**	**132**
L_COMMTEMPLATE	DELETE	OWNERID=132,LANGCODE=DE	12/4/12 9:08 PM	WILSON	COMMTEMPLATE	132

OK Cancel

Figure 15-25 Change tracking delete action records

Figure 15-26 shows the effect on the migration collection of the deletion of the TESTDE communication template. The collection record is deleted.



Configurations | Filter | 1 - 1 of 1

Application	Primary Keys	Object Structure	Create By	Create Date	Create Mode
CONFIGUR	**OBJECTNAME=SR**	DMMAXOBJE	**WILSON**	**12/4/12 6:29 PM**	**AUTO**

Download Exported Configurations

Figure 15-26 Effect of delete action on a migration collection

15.7 Admin mode

The Tivoli's process automation engine requires exclusive access to the back-end database to configure structural changes to its objects. In the past, this configuration took place after the application server was shut down with the purpose of restricting user access and applying the configuration changes successfully.

With Admin mode, the application now can restrict the user access and make the corresponding database configuration changes without the need to shut down the application server.

In Admin mode, cron jobs are suspended, and any event listener stops until it is turned off.

This operating mode also affects the Migration Manager. The Migration Manager requires that Admin mode is turned on during the deployment of configuration changes that are made to database objects, which are also known as *structural changes*. The Migration Manager also requires that Admin mode is turned on when a package is deployed that includes the MAXAPPS object or application designer changes.

Therefore, depending on the type of package that is deployed, the Admin mode is needed so that the changes are reflected on the user's session. However, it is always a good idea to turn on this mode when package deployment windows are used.

There are other considerations when you are working on a multiple server or cluster environment. For more information, see 15.5, "Clustered environment considerations" on page 368.

15.8 Migrating hierarchical data

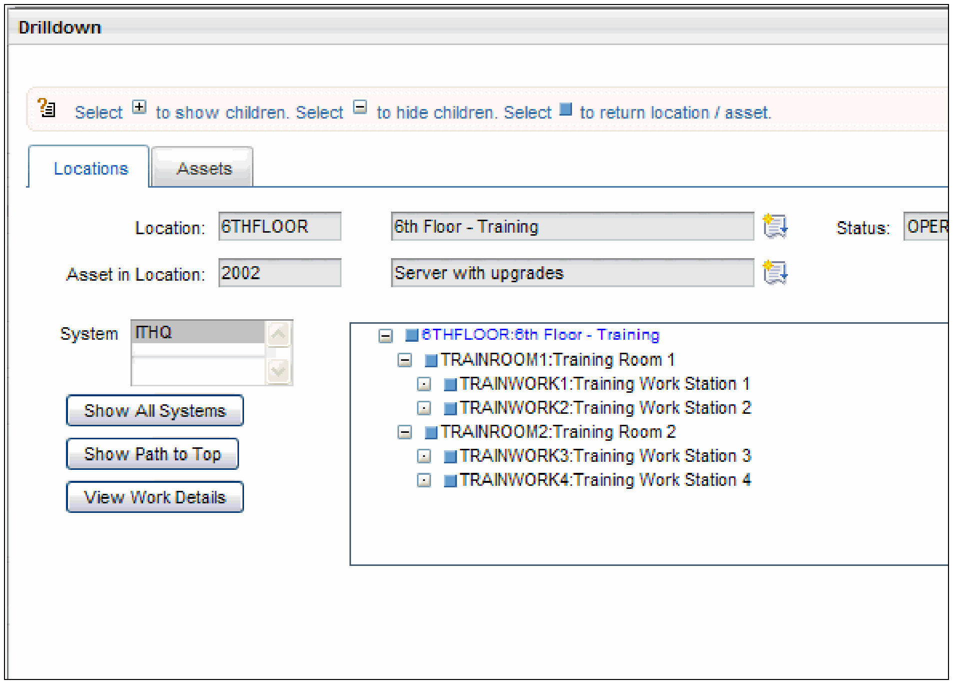
Historically, the migration of hierarchical data was difficult. Today, this problem is solved. This section provides a scenario for the migration of a location hierarchy. There is a unique situation that occurs with hierarchical data. The object structure must be self-referencing and it must include a database reference to itself to be self-referencing.

The following steps are necessary to create a migration for hierarchical data:

1. Identify the hierarchical data.
2. Identify the object structure to use as a template.
3. Create a self-referencing relationship.
4. Duplicate the object structure that was identified in step 2.
5. Migrate the object structure.
6. Migrate the data.

Identifying the hierarchical data for migration

In this scenario, the construction of the training facility on the 6th floor is complete. Assets now must be placed into production and you must move the hierarchy from the test environment into production. Figure 15-27 shows the location hierarchy in the drill-down view.


Drilldown

Select ⊞ to show children. Select ⊟ to hide children. Select ■ to return location / asset.

Locations | Assets

Location: 6THFLOOR 6th Floor - Training Status: OPER

Asset in Location: 2002 Server with upgrades

System ITHQ

Show All Systems

Show Path to Top

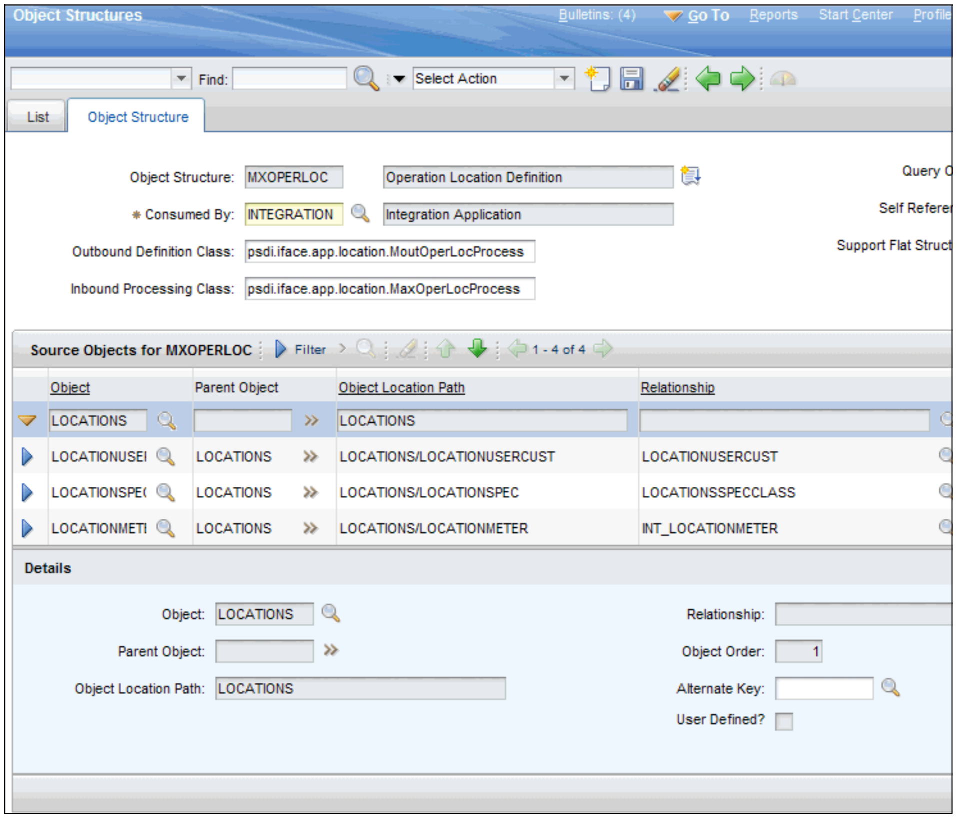
View Work Details

```
⊟ ■6THFLOOR:6th Floor - Training
   ⊟ ■TRAINROOM1:Training Room 1
      ⊡ ■TRAINWORK1:Training Work Station 1
      ⊡ ■TRAINWORK2:Training Work Station 2
   ⊟ ■TRAINROOM2:Training Room 2
      ⊡ ■TRAINWORK3:Training Work Station 3
      ⊡ ■TRAINWORK4:Training Work Station 4
```

Figure 15-27 Example of location hierarchy

Identifying the object structure that is required as a template

After you identify the type of hierarchical data to migrate, you must identify the appropriate object structure to use as a template. Because we are moving the location data, we must use the MXOPERLOC object structure as a template for the new object structure, as shown in Figure 15-28.



Object Structure: MXOPERLOC Operation Location Definition

* Consumed By: INTEGRATION Integration Application

Outbound Definition Class: psdi.iface.app.location.MoutOperLocProcess

Inbound Processing Class: psdi.iface.app.location.MaxOperLocProcess

Object	Parent Object	Object Location Path	Relationship
LOCATIONS		LOCATIONS	
LOCATIONUSEI	LOCATIONS	LOCATIONS/LOCATIONUSERCUST	LOCATIONUSERCUST
LOCATIONSPE(	LOCATIONS	LOCATIONS/LOCATIONSPEC	LOCATIONSSPECCLASS
LOCATIONMETI	LOCATIONS	LOCATIONS/LOCATIONMETER	INT_LOCATIONMETER

Object: LOCATIONS

Parent Object:

Object Location Path: LOCATIONS

Relationship:

Object Order: 1

Alternate Key:

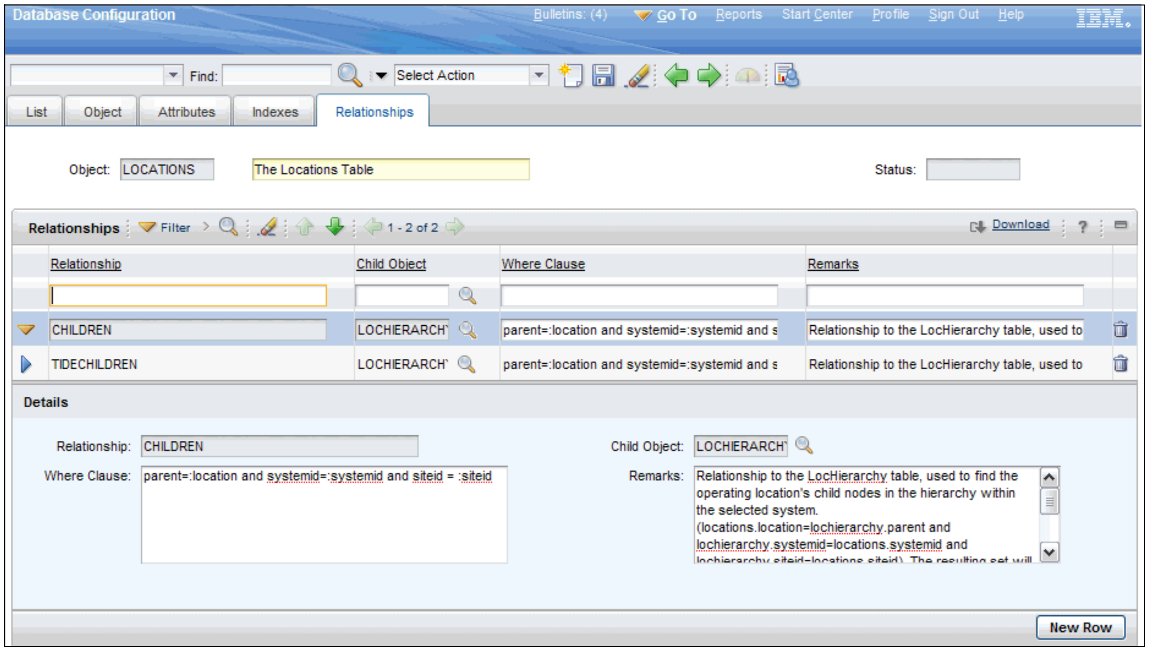
User Defined?

Figure 15-28 The MXOPERLOC object structure

Now that we identified the object structure, we can determine what the relationship must be to create the object structure.

Creating the new self-referencing relationship

After you determine the nature of the relationship (as dictated by the template object structure), you must create the relationship by using the Database Configuration application. In this case, because we use location hierarchy data, we can use the existing CHILDREN relationship to create a relationship for self-referencing. Figure 15-29 shows the CHILDREN relationship.



Object: LOCATIONS The Locations Table

Status:

Relationships

Relationship	Child Object	Where Clause	Remarks
CHILDREN	LOCHIERARCH'	parent=:location and systemid=:systemid and s	Relationship to the LocHierarchy table, used to
TIDECHILDREN	LOCHIERARCH'	parent=:location and systemid=:systemid and s	Relationship to the LocHierarchy table, used to

Details

Relationship: CHILDREN

Child Object: LOCHIERARCH'

Where Clause: parent=:location and systemid=:systemid and siteid = :siteid

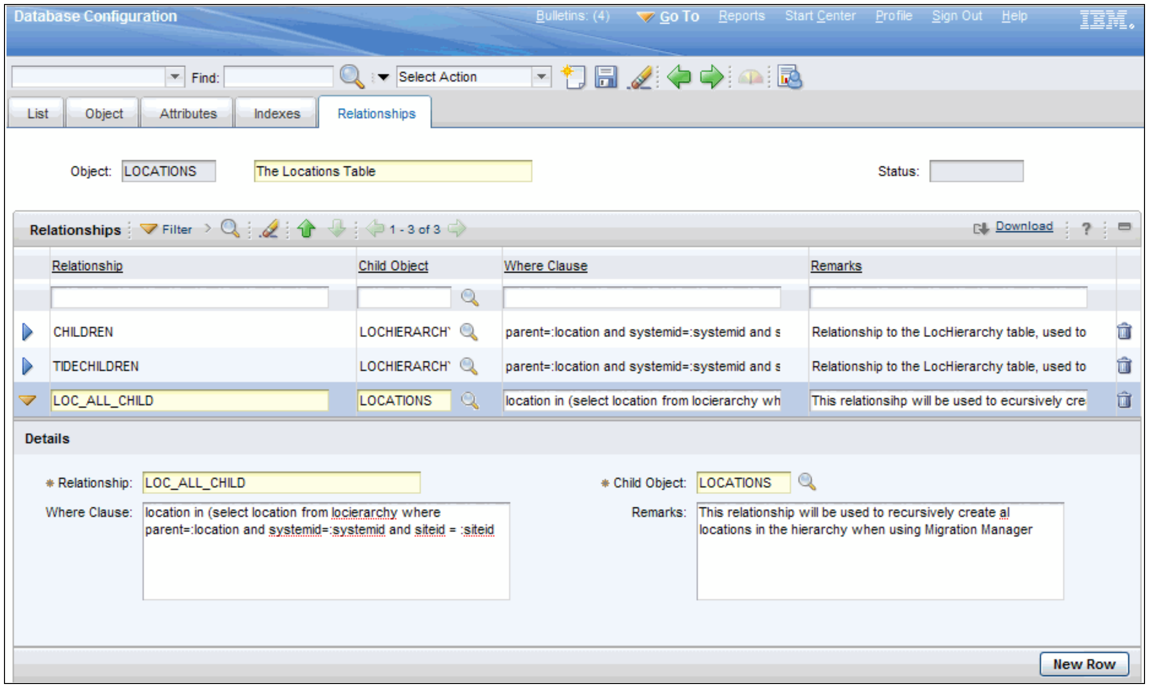
Remarks: Relationship to the LocHierarchy table, used to find the operating location's child nodes in the hierarchy within the selected system. (locations.location=lochierarchy.parent and lochierarchy.systemid=locations.systemid and lochierarchy.siteid=locations.siteid). The resulting set will

Figure 15-29 CHILDREN relationship for location

Complete the following steps to set up a relationship:

1. Click **New Row**. Enter a new relationship name. In the Child Object field, enter `LOCATIONS` (as shown in Figure 15-30). Enter a new WHERE clause that selects all of the children for the current location, for example:

```
location in (select location from lochierarchy where parent=
:location and systemid = :systemid and siteid = :siteid
```


Database Configuration

Bulletins: (4) | Go To | Reports | Start Center | Profile | Sign Out | Help

Find: | Select Action

List | Object | Attributes | Indexes | Relationships

Object: LOCATIONS | The Locations Table | Status:

Relationships | Filter | 1 - 3 of 3 | Download

Relationship	Child Object	Where Clause	Remarks
CHILDREN	LOCHIERARCH'	parent=:location and systemid=:systemid and s	Relationship to the LocHierarchy table, used to
TIDECHILDREN	LOCHIERARCH'	parent=:location and systemid=:systemid and s	Relationship to the LocHierarchy table, used to
LOC_ALL_CHILD	LOCATIONS	location in (select location from locierarcy wh	This relationsihp will be used to ecursively cre

Details

* Relationship: LOC_ALL_CHILD

Where Clause: location in (select location from locierarchy where parent=:location and systemid=:systemid and siteid = :siteid

* Child Object: LOCATIONS

Remarks: This relationship will be used to recursively create al locations in the hierarchy when using Migration Manager

New Row

Figure 15-30 New self-referencing relationship

2. Enter remarks that define the purpose of the relationship.

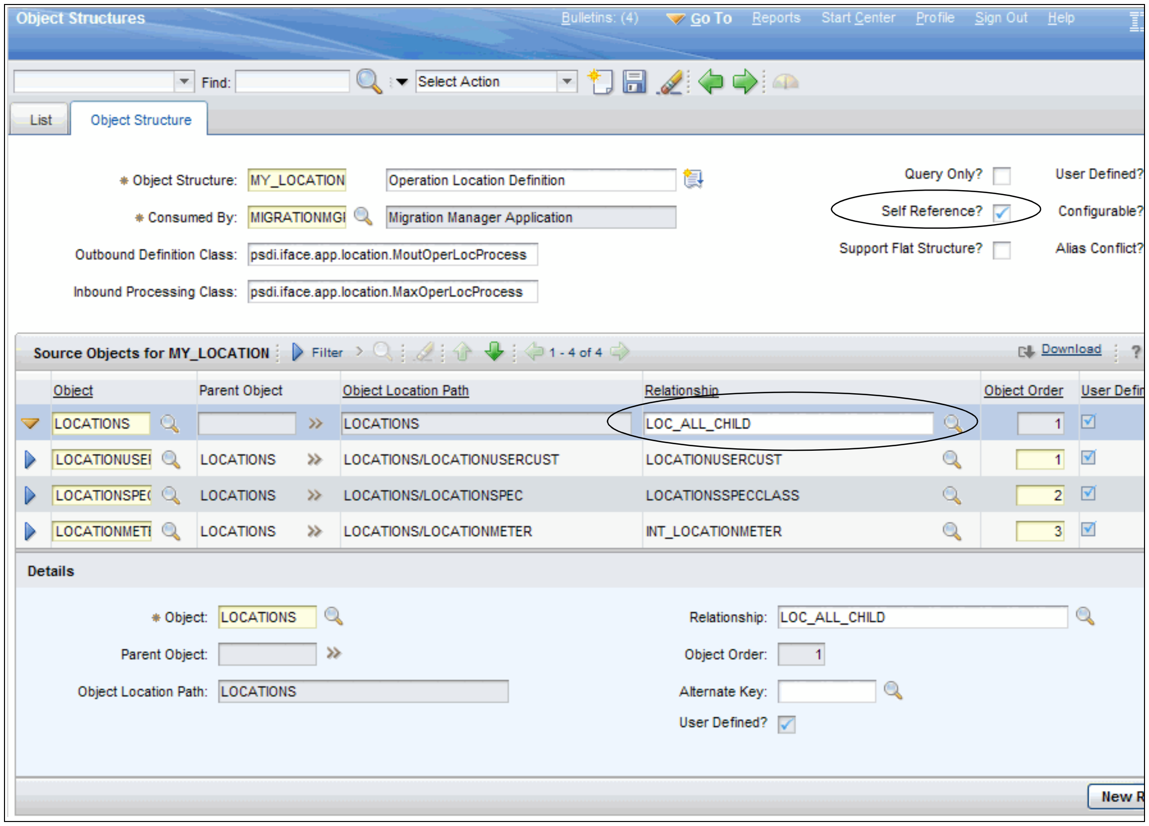
Relationships: Because the child table in the relationship is the same as the parent, for every record that is selected by using the Outbound processing class, every child also is selected. As a result, the processing class recursively selects every child in the tree.

3. Save the record and return to the Object Structure application.

Creating the object structure

Now that the new relationship is in place, you can duplicate the object structure that was identified in Figure 15-28 on page 387 by completing the following steps:

1. Click **Select Action** → **Duplicate Object Structure**. A new object structure is shown that is identical to the MXOPERLOC object structure.
2. Enter the name of the new object structure, as shown in Figure 15-31.



Object Structure: MY_LOCATION — Operation Location Definition
Consumed By: MIGRATIONMGI — Migration Manager Application
Outbound Definition Class: psdi.iface.app.location.MoutOperLocProcess
Inbound Processing Class: psdi.iface.app.location.MaxOperLocProcess
Query Only? ☐ User Defined?
Self Reference? ✓ Configurable?
Support Flat Structure? ☐ Alias Conflict?

Source Objects for MY_LOCATION — Filter — 1 - 4 of 4 — Download

Object	Parent Object	Object Location Path	Relationship	Object Order	User Defin
LOCATIONS		LOCATIONS	LOC_ALL_CHILD	1	✓
LOCATIONUSEI	LOCATIONS	LOCATIONS/LOCATIONUSERCUST	LOCATIONUSERCUST	1	✓
LOCATIONSPE(	LOCATIONS	LOCATIONS/LOCATIONSPEC	LOCATIONSSPECCLASS	2	✓
LOCATIONMETI	LOCATIONS	LOCATIONS/LOCATIONMETER	INT_LOCATIONMETER	3	✓

Details

Object: LOCATIONS
Parent Object:
Object Location Path: LOCATIONS
Relationship: LOC_ALL_CHILD
Object Order: 1
Alternate Key:
User Defined? ✓

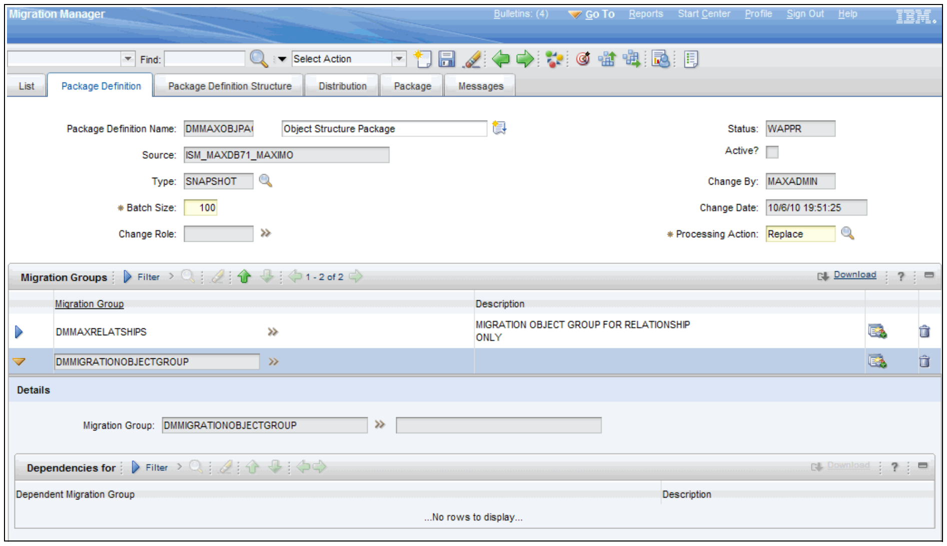
Figure 15-31 New object structure for the migration service

We selected the **Self Reference?** option and used the relationship that we created.

Migrating the object structure

Now that you created the object structure, you must create a migration package that migrates the object structure and the relationship.

You can create a set of migration groups or you can use the predefined migrating groups. If you use the predefined migration groups, you must set a number of SQL WHERE clause values to 1=2 to limit the records that are migrated, as shown in Figure 15-32.



Package Definition Name: DMMAXOBJPA(Object Structure Package
Source: ISM_MAXDB71_MAXIMO
Type: SNAPSHOT
Batch Size: 100
Change Role:
Status: WAPPR
Active?
Change By: MAXADMIN
Change Date: 10/6/10 19:51:25
Processing Action: Replace

Migration Groups | Filter | 1 - 2 of 2

Migration Group	Description
DMMAXRELATSHIPS	MIGRATION OBJECT GROUP FOR RELATIONSHIP ONLY
DMMIGRATIONOBJECTGROUP	

Details

Migration Group: DMMIGRATIONOBJECTGROUP

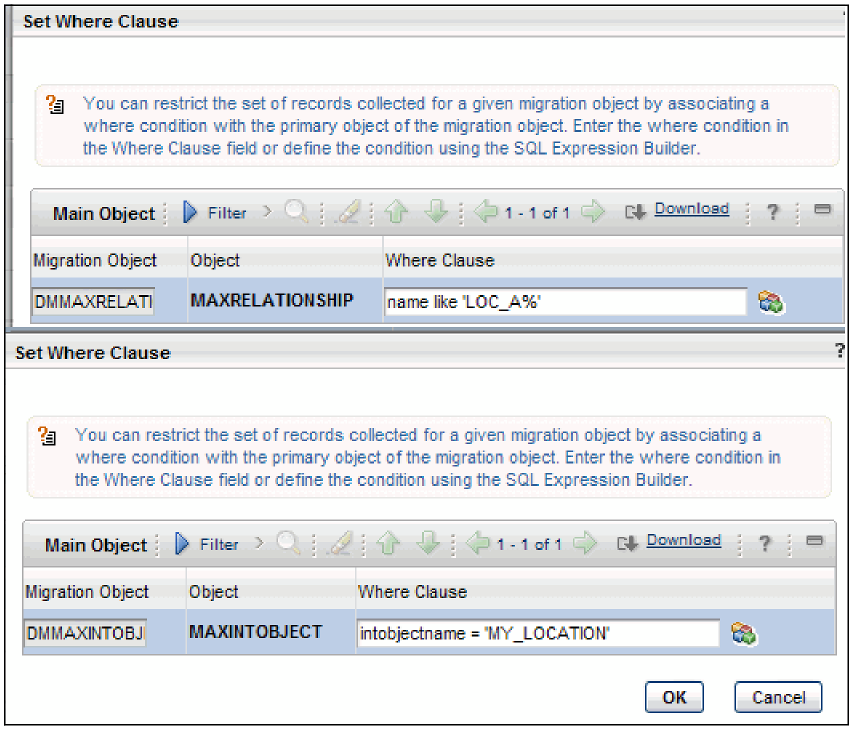
Dependencies for | Filter

Dependent Migration Group	Description

...No rows to display...

Figure 15-32 New migration package for moving the new object structure

In this example, you can see that the relationship and the object structure are the only migration objects in the group. Similarly, the groups are limited by the WHERE clause, as shown in Figure 15-33.



You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

Migration Object	Object	Where Clause
DMMAXRELATI	MAXRELATIONSHIP	name like 'LOC_A%'

You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

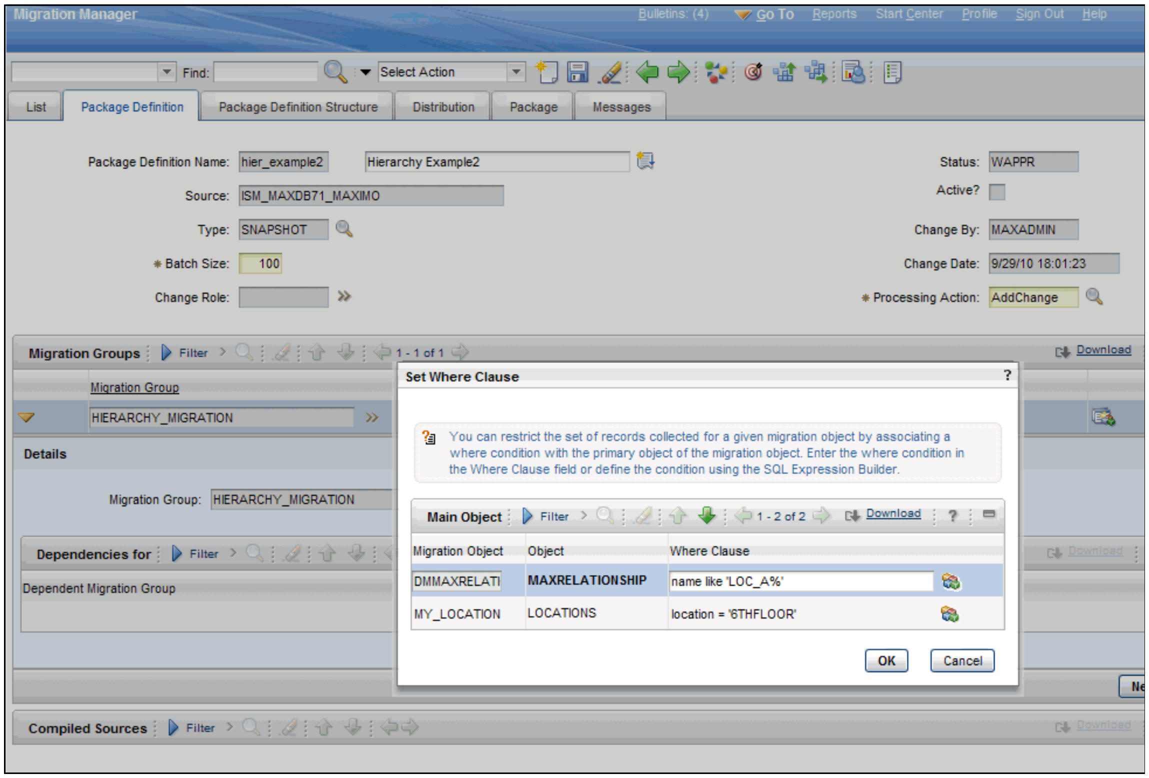
Migration Object	Object	Where Clause
DMMAXINTOBJ	MAXINTOBJECT	intobjectname = 'MY_LOCATION'

Figure 15-33 Setting the WHERE clause for several groups

We find it easier to create migration groups with limited object structures and set the SQL WHERE clause for the limited migration groups. This approach speeds up the package processing time.

Migrating the object structure

You can now migrate the location hierarchy by creating the package. The migration group features only a single object structure, as shown in Figure 15-34.


Package Definition Name: hier_example2 Hierarchy Example2

Source: ISM_MAXDB71_MAXIMO

Type: SNAPSHOT

Batch Size: 100

Change Role:

Status: WAPPR

Active?

Change By: MAXADMIN

Change Date: 9/29/10 18:01:23

Processing Action: AddChange

Migration Group: HIERARCHY_MIGRATION

Set Where Clause

You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

Migration Object	Object	Where Clause
DMMAXRELATI	MAXRELATIONSHIP	name like 'LOC_A%'
MY_LOCATION	LOCATIONS	location = '6THFLOOR'

Figure 15-34 Location hierarchy migration example

Now process the migration package.

15.9 Migration Manager comparisons

In Chapter 4, “Migration Manager Comparison” on page 75, we introduced the comparison feature of Migration Manager. The comparison feature was added in response to customer needs to accelerate migration by using the most efficient techniques possible. By comparing data at the table level between source and target, differences in values can be found rapidly. These differences can be fed into the Migration Manager toolset to generate a collection and from the collection, generate a package definition. Upon successful migration of the package into target, the two environments are rendered identical from the system configuration perspective. This capability of the Migration Manager also puts Tivoli’s process automation engine-based products on par with similar tools offered by other software vendors.

Important: Throughout this section, the term *differences* is used to represent a range of comparison results that are identified by Migration Manager. One of the comparison result types that is returned is called *DIFFERENCE*. DIFFERENCE specifically represents the situation where a system configuration exists in source and target but values for one or more columns of a system configuration record are different in source and in target. This distinction helps implementers better understand the functionality and its usage.

15.9.1 Primary use cases for comparison

The following major use cases drive a comparison approach:

- When source and target product environments are in place and the source is used actively to implement system configurations, the most efficient migration approach is to compare source and target and migrate the differences.
- After a migration is completed, comparison can be run from the source or the target to verify that there is now a match between the system configurations of the source and target environments. This verification approach is faster than manual application-by-application, configuration-by-configuration verification.

15.9.2 Pre-requisites for comparison

The following prerequisites must be met to run comparison:

- A database connection must be defined by using the Manage Targets functionality of Migration Manager. The Comparison function obtains a connection to the remote database by using Tivoli's process automation engine's connection APIs.
- Base language for the two participating product environments must be the same.
- Relational databases for the two participating product environments must be the same. Comparison does not run if one of the product environments uses Oracle database and the other uses DB2, for example.
- Admin mode should be turned off. If a Comparison job was submitted when Admin mode was on in the initiating product environment, the job remains in SUBMITTED state. After Admin mode is turned off, previously submitted jobs run.

15.9.3 Comparison restrictions

The following restrictions apply to any comparison that is run by Migration Manager:

- Relational database metadata information is not compared. For example, DB2 system catalog views, such as SYSCAT.TABLES, which holds information for tables and views in the database.
- From the Tivoli's process automation engine data dictionary, multi-language, eAudit, long description tables are not compared.
- Non-persistent business objects are not compared. Only Tivoli's process automation engine data dictionary tables and views are compared.
- Tivoli's process automation engine data dictionary attributes with data types of CLOB, BLOB, CRYPTO, CRYPTOX, LONGALN, DATE, and DATETIME are not compared.
- Tivoli's process automation engine data dictionary attributes, such as CREATEBY, CREATEDATE, STATUSDATE, SENDERSYSID, CHANGEBY, CHANGEDATE, LASTUPDATED, PASSWORD, ENCRYPTEDVALUE, and ROWSTAMP are not compared.
- If a snapshot package with filters in the form of SQL criteria is used as the basis for comparison, the SQL criteria are not used in comparison.
- There is no facility to specify or control business objects or attributes to be skipped for comparison.

15.9.4 Running comparison

Comparison can be run after the prerequisites are met. Comparison is run from the Migration Manager application user interface. A comparison job is prepared for running on the server-side. At any time, **Refresh Job Status** can be clicked to view the progress. Usually, progress is indicated by the increasing count of the number of records that were compared. Upon completion, the comparison job state is set to COMPLETED.

15.9.5 Processing comparison results

Comparison results are stored in the database from where the comparison was run. Storing the results enables implementers to examine the various comparison outcomes and examine the overall state of system configurations in the two participating product environments. Comparison results can be reported on eliminating or reducing the need for implementation teams to author documents that enumerate the changes.

A migration collection can be created or the existing migration collection can be used to receive a subset of the comparison result records (LOCALONLY and DIFFERENCE). This collection can then be reviewed, trimmed, or expanded until it is deemed complete and ready for migration.

15.9.6 Limitations

From our use of the Comparison functionality, we observed the following limitations, which we expect to address in the upcoming fix packs or releases to make the functionality even more usable:

- Running comparison against the SCRIPTCFG migration group for Oracle databases might result in an error. The following error message is displayed in the Comparison Error window:

  ```
  java.sql.SQLException:Exhausted Resultset

  at
  oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMappin
  g.java:70)
  ```

- Comparison skips tables or views when the data dictionary definition of the corresponding business object includes attributes that are designated primary key columns but are also marked as not required. For example, comparison skips the ESCREFPOINT table during comparison of escalation system configurations. The ESCALATION attribute of ESCREFPOINT is marked as a primary key column but it is also marked as not required.

The effect of skipping tables or views is a partial comparison or even a set of misleading results. For example, because the ESCREFPOINT table is skipped, different values in source and target for the EVENTCONDITION attribute are not taken into account for the comparison.

Table 15-8 lists the object structures, business objects, and attributes that are affected by this limitation.

Table 15-8 Object structures and business objects affected by comparison limitation

Object Structure	Business Object
DMACTIONGROUP	ACTIONGROUP
DMCLASSIFICATION	CLASSSPEC, CLASSSPECUSEWITH
DMCOMMTEMPLATE	COMMTMPLTSENDTO
DMDEFGLCONFIGURE	GLCONFIGURE
DMESCALATION	ESCREFPOINT
DMINTERACTION	MAXINTMAPPING
DMMAXDOMAIN	ALNDOMAIN, CROSSOVERDOMAIN, MAXTABLEDOMAIN, NUMERICDOMAIN, NUMRANGEDOMAIN, SYNONYMDOMAIN MAXDOMVALCOND
DMMAXEXTSYSTEM	MAXEXTBOOLVAL, MAXEXTCTLVAL, MAXEXTOVER, MAXEXTXREFVAL, MAXLISTOVERVAL, MAXXREFOVERVAL
DMMAXGROUP	SECURITYRESTRICT
DMMAXIFACECONTROL	MAXCONTROLVALUE
DMMAXLAUNCHENTRY	MAXLECONTEXT
DMMAXMENU	MAXMENU
DMMAXOBJECTCFG	AUTOKEY
DMMAXPROP	MAXPROPVALUE
DMMAXVARS	MAXVARS
DMPACKAGE	DMPKGSTATUS

Object Structure	Business Object
DMPERSON	PHONE, SMS
DMPERSONGROUP	PERSONGROUPTEAM
DMREPORT	REPORTAPPAUTH, REPORTLABEL, REPORTLOKUP
DMTKTEMPLATE	TKTEMPLATESPEC,TKTEMPLTACTYSPEC

15.9.7 Using comparisons effectively

In this section, we describe some of the techniques that can improve the effectiveness of the comparison approach.

Use a dedicated package definition from which to run comparisons. This definition is not a definition that is used to create packages with system configuration content. Leave the package definition with WAPPR status. This centralized approach includes the following advantages:

- Avoid ad hoc comparison approaches where comparison results are spread over multiple package definitions. Implementers use the designated package definition.
- Add or remove migration groups to or from the package definition as the need arises to increase or decrease the scope of comparisons.
- After each comparison completes, run ad hoc reports against the comparison results and share the report outputs among members of the implementation team for review.
- After comparison results are reviewed, identify the migration collection into which system configuration records that represent outcomes of LOCALONLY and DIFFERENCE can be copied into. After the system configurations are placed into a collection, the collection can be managed in preparation for migration.
- Periodically purge the underlying comparison data tables by deleting comparison jobs from the Migration Manager user interface. Purging comparison data ensures control over the potential rapid data growth in these tables.

Figure 15-35 shows one potential approach to the use of comparisons to drive migration. As development iterations unfold, each iteration's system configurations are uncovered by comparing the source (development) environment with the target (production) environment. The differences are placed into a dedicated migration collection as the starting point for migration.

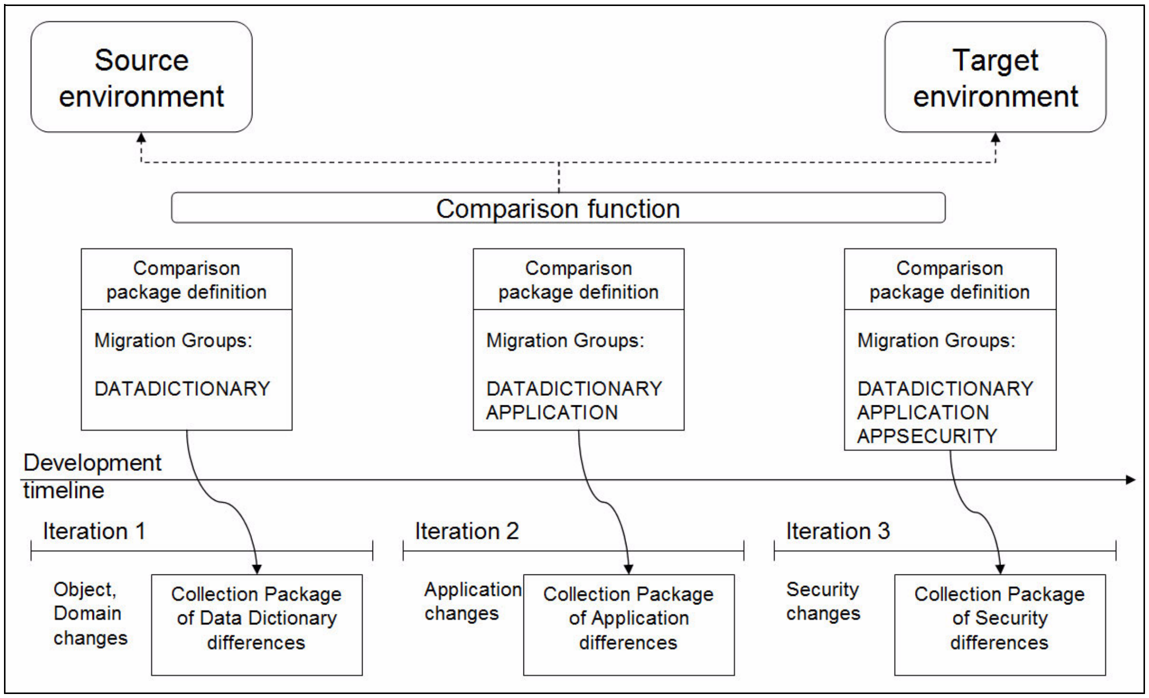


Figure 15-35 Comparison usage scenario

15.10 Error correction and package reprocessing

Until the Tivoli's process automation engine release 7.5, Migration Manager did not offer an error correction and reprocessing capability. If the deployment of a migration package resulted in an error because of data validation, deployment was stopped. To resolve the error, the implementer had to repackage the failing system configuration in the source environment. The new package was distributed and deployed. It was likely that several valid system configurations were unnecessarily updated because of the repackaging.

With error correction and package reprocessing, this costly approach to managing failing packages can be avoided and migration velocity maintained. A pause-correct-resume pattern can be implemented. Upon encountering a deployment error, Migration Manager stops.

From an implementation perspective, the use of error correction and package reprocessing can be characterized as a pause-correct-resume cycle. Figure 15-36 on page 401 shows this cycle and highlights the history of error information that the Migration Manager accumulates over time for a package.

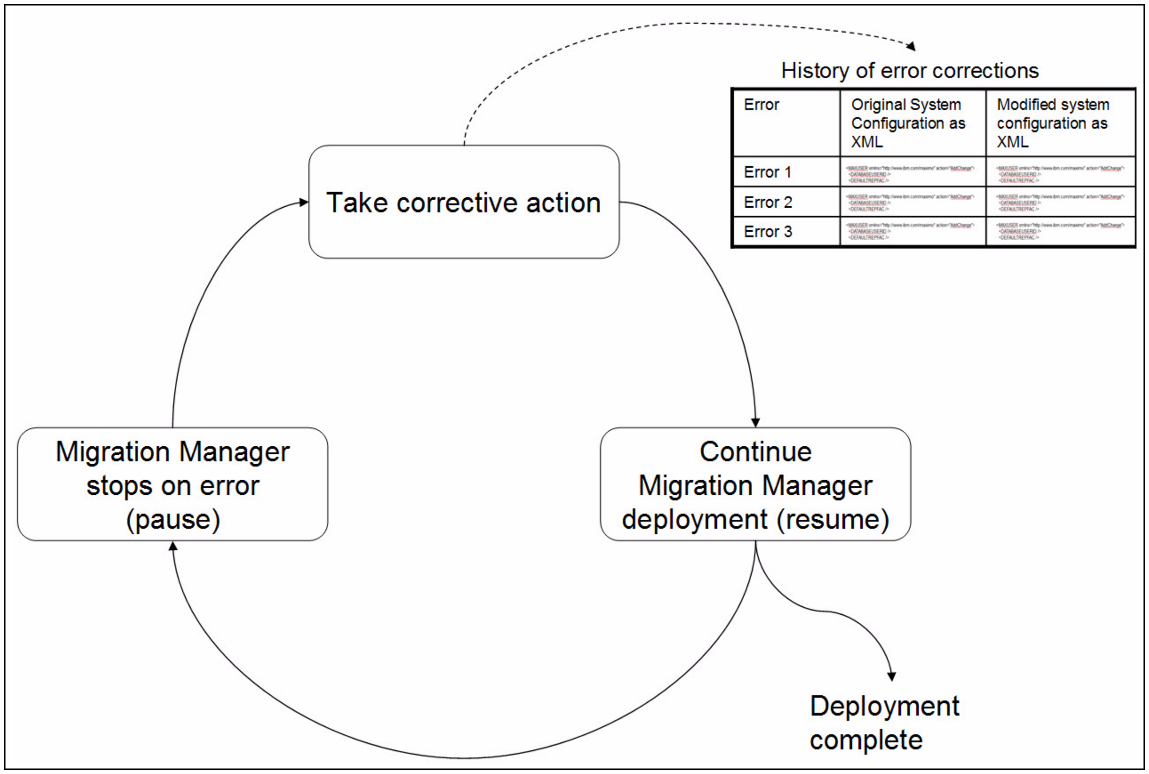

History of error corrections
Error
Original System Configuration as XML
Modified system configuration as XML
Error 1
Error 2
Error 3
Take corrective action
Migration Manager stops on error (pause)
Continue Migration Manager deployment (resume)
Deployment complete

Figure 15-36 Pause-correct-resume cycle to manage deployment errors

Upon encountering an error, package deployment is stopped. An error record is written out by Migration Manager. An implementer can treat the situation as a pause and determine the cause of the error and the corrective action to be taken. Corrective action can consist of one or both of the following actions:

- If the deployment error is the result of invalid data in the system configuration, edit the failing system configuration through its XML representation. Save the edited XML by using Migration Manager. The updated XML data is written back into the error record.
- If the deployment error is the result of a dependency of the system configuration on another configuration or setting in the target environment, update that dependent system configuration or change the setting.

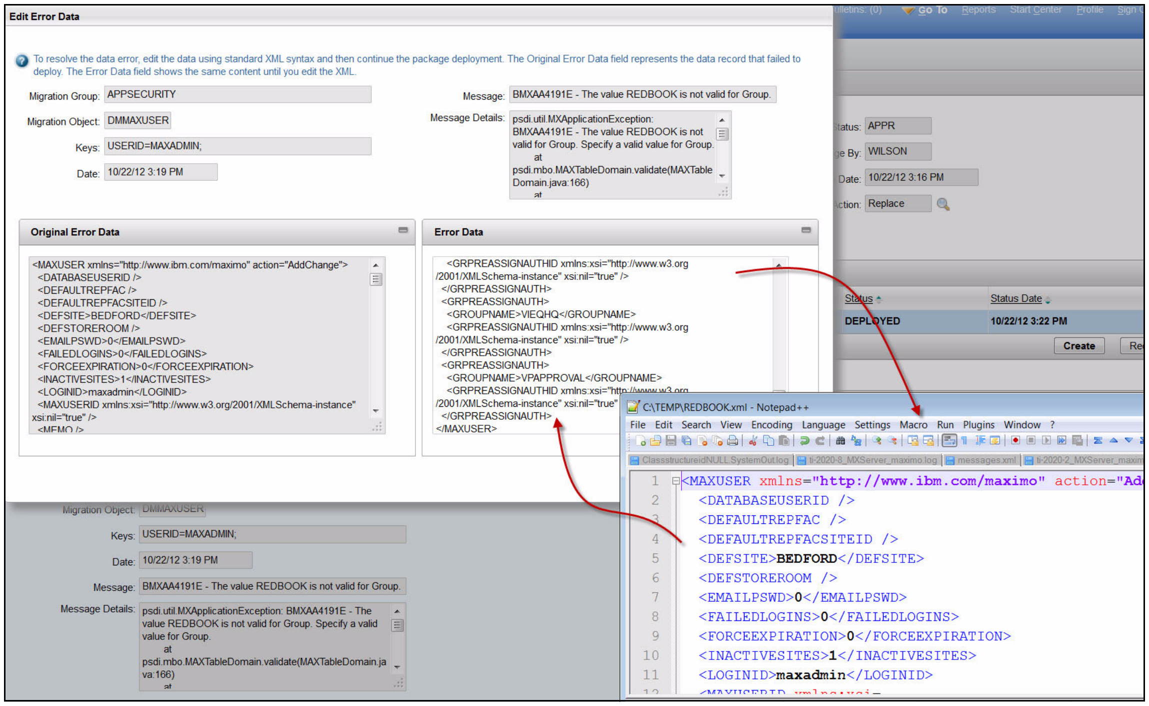
After corrective action is taken, the implementer can return to Migration Manager and continue the deployment by clicking **Continue Deployment** in the Packages tab of the application. Deployment resumes from the point of failure within the package. The point of failure is the specific package XML that contained the invalid system configuration.

15.10.1 Error correction and reprocessing user interface

The Migration Manager application provides the user interface, with which you can use error correction and reprocessing. The Deployment Data Errors subtab becomes visible within the Packages tab of the application only if the deployment error is triggered as a result of processing data for a system configuration from within the package. The Edit Error Data within the Deployment Data Errors subtab enables access to the Edit Error Data error correction window. This window offers an XML-representation of the failing XML. The text editor does not include any syntax coloring and XML validation capabilities.

Best practice: Use an external XML editor to review and change a failing system configuration XML. Copy the text from the Error Data field of the Edit Error Data window into an external editor. After the text is edited, paste the updated XML data into the Error Data field.

Figure 15-37 shows the Edit Error Data window from Migration Manager and the use of a more XML-friendly text editor to review and edit the XML representation.


Edit Error Data
Original Error Data
Error Data
C:\TEMP\REDBOOK.xml - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
ClassstructureidNULL.SystemOut.log
ti-2020-8_MXServer_maximo.log
messages.xml
Create

To resolve the data error, edit the data using standard XML syntax and then continue the package deployment. The Original Error Data field represents the data record that failed to deploy. The Error Data field shows the same content until you edit the XML.

Migration Group: APPSECURITY
Migration Object: DMMAXUSER
Keys: USERID=MAXADMIN;
Date: 10/22/12 3:19 PM
Message: BMXAA4191E - The value REDBOOK is not valid for Group.
Message Details: psdi.util.MXApplicationException: BMXAA4191E - The value REDBOOK is not valid for Group. Specify a valid value for Group.
at
psdi.mbo.MAXTableDomain.validate(MAXTableDomain.java:166)
at

Original Error Data:

```
<MAXUSER xmlns="http://www.ibm.com/maximo" action="AddChange">
  <DATABASEUSERID />
  <DEFAULTREPFAC />
  <DEFAULTREPFACSITEID />
  <DEFSITE>BEDFORD</DEFSITE>
  <DEFSTOREROOM />
  <EMAILPSWD>0</EMAILPSWD>
  <FAILEDLOGINS>0</FAILEDLOGINS>
  <FORCEEXPIRATION>0</FORCEEXPIRATION>
  <INACTIVESITES>1</INACTIVESITES>
  <LOGINID>maxadmin</LOGINID>
  <MAXUSERID xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
  <MEMO />
```

Error Data:

```
   <GRPREASSIGNAUTHID xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance" xsi:nil="true" />
  </GRPREASSIGNAUTH>
  <GRPREASSIGNAUTH>
   <GROUPNAME>VIEQHQ</GROUPNAME>
   <GRPREASSIGNAUTHID xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance" xsi:nil="true" />
  </GRPREASSIGNAUTH>
  <GRPREASSIGNAUTH>
   <GROUPNAME>VPAPPROVAL</GROUPNAME>
   <GRPREASSIGNAUTHID xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance" xsi:nil="true"
  </GRPREASSIGNAUTH>
</MAXUSER>
```

Status: APPR
Change By: WILSON
Date: 10/22/12 3:16 PM
Action: Replace

Status	Status Date
DEPLOYED	10/22/12 3:22 PM

Migration Object: DMMAXUSER
Keys: USERID=MAXADMIN;
Date: 10/22/12 3:19 PM
Message: BMXAA4191E - The value REDBOOK is not valid for Group.
Message Details: psdi.util.MXApplicationException: BMXAA4191E - The value REDBOOK is not valid for Group. Specify a valid value for Group.
at
psdi.mbo.MAXTableDomain.validate(MAXTableDomain.java:166)
at

```
<MAXUSER xmlns="http://www.ibm.com/maximo" action="Ad
  <DATABASEUSERID />
  <DEFAULTREPFAC />
  <DEFAULTREPFACSITEID />
  <DEFSITE>BEDFORD</DEFSITE>
  <DEFSTOREROOM />
  <EMAILPSWD>0</EMAILPSWD>
  <FAILEDLOGINS>0</FAILEDLOGINS>
  <FORCEEXPIRATION>0</FORCEEXPIRATION>
  <INACTIVESITES>1</INACTIVESITES>
  <LOGINID>maxadmin</LOGINID>
```

Figure 15-37 Correcting error XML with an external editor

15.10.2 Migration error correction examples

We show the use of error correction and reprocessing with the following examples:

- Upon the migration of a package that contains a data dictionary business object definition, deployment fails with the following error:

  ```
  BMXAA0686E - This object is audit enabled, but audit table name was
  not specified.
  ```

  This error can be corrected by editing the failing XML. Providing a value to the EAUDITTBNAME XML tag and resubmitting the edited XML to Migration Manager enables the implementer to continue the deployment.

- Upon the migration of a package that contains a user configuration, the deployment fails and the following error is shown:

  ```
  BMXAA4191E - The value REDBOOK is not valid for Group. Specify a
  valid value for Group.
  ```

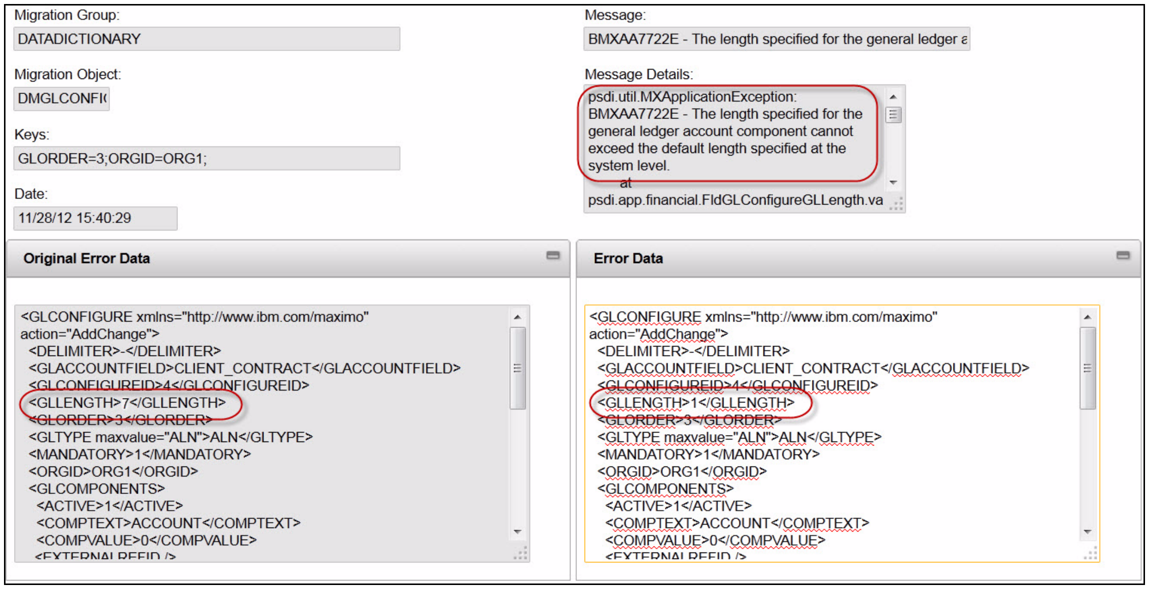
  This error can be resolved by editing the failing XML. References to the REDBOOK group can be removed from the XML and edited XML can be resubmitted to the Migration Manager. A catch up action is needed to migrate the missing REDBOOK security group and add the user into that group. A separate package can be used to migrate the REDBOOK security group.

- Upon the migration of a package that contains a General Ledger configuration, the deployment fails and the following error is shown:

  ```
  BMXAA7722E - The length specified for the general ledger account
  component cannot exceed the default length specified at the system
  level.
  ```

  This error can be corrected by editing the failing XML. The length that is specified in the GLLENGTH XML tag can be reduced to the allowed length. Resubmitting the edited XML to Migration Manager enables the implementer to continue the deployment.

Figure 15-38 shows the edit that was made to the GLCONFIGURE XML through the Edit Error Data window in response to the BMXAA7722E error message.


Migration Group: DATADICTIONARY

Migration Object: DMGLCONFI

Keys: GLORDER=3;ORGID=ORG1;

Date: 11/28/12 15:40:29

Message: BMXAA7722E - The length specified for the general ledger a

Message Details:

```
psdi.util.MXApplicationException:
BMXAA7722E - The length specified for the
general ledger account component cannot
exceed the default length specified at the
system level.
        at
psdi.app.financial.FldGLConfigureGLLength.va
```

Original Error Data

```
<GLCONFIGURE xmlns="http://www.ibm.com/maximo"
action="AddChange">
  <DELIMITER>-</DELIMITER>
  <GLACCOUNTFIELD>CLIENT_CONTRACT</GLACCOUNTFIELD>
  <GLCONFIGUREID>4</GLCONFIGUREID>
  <GLLENGTH>7</GLLENGTH>
  <GLORDER>3</GLORDER>
  <GLTYPE maxvalue="ALN">ALN</GLTYPE>
  <MANDATORY>1</MANDATORY>
  <ORGID>ORG1</ORGID>
  <GLCOMPONENTS>
    <ACTIVE>1</ACTIVE>
    <COMPTEXT>ACCOUNT</COMPTEXT>
    <COMPVALUE>0</COMPVALUE>
    <EXTERNALREFID />
```

Error Data

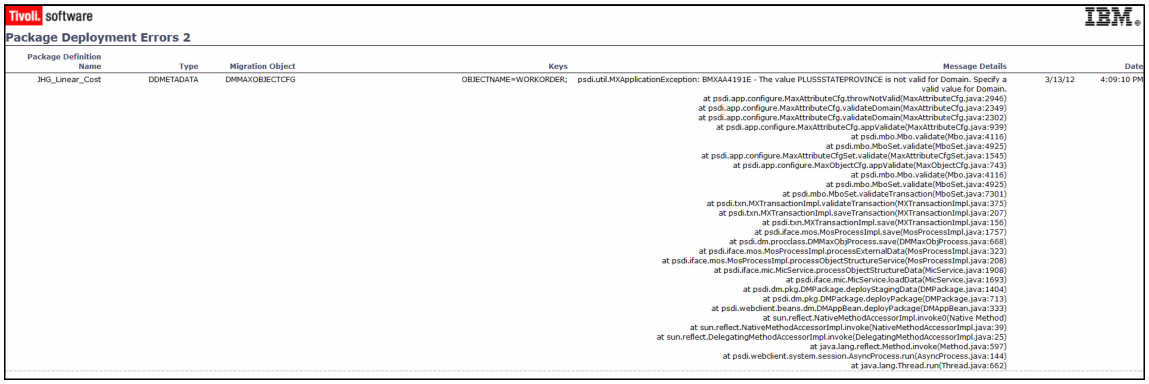
```
<GLCONFIGURE xmlns="http://www.ibm.com/maximo"
action="AddChange">
  <DELIMITER>-</DELIMITER>
  <GLACCOUNTFIELD>CLIENT_CONTRACT</GLACCOUNTFIELD>
  <GLCONFIGUREID>4</GLCONFIGUREID>
  <GLLENGTH>1</GLLENGTH>
  <GLORDER>3</GLORDER>
  <GLTYPE maxvalue="ALN">ALN</GLTYPE>
  <MANDATORY>1</MANDATORY>
  <ORGID>ORG1</ORGID>
  <GLCOMPONENTS>
    <ACTIVE>1</ACTIVE>
    <COMPTEXT>ACCOUNT</COMPTEXT>
    <COMPVALUE>0</COMPVALUE>
    <EXTERNALREFID />
```

Figure 15-38 Editing bad data by using the Edit Error Data window

15.10.3 Reporting on package errors

Previously in this chapter, we noted that the deployment errors that were encountered for a package are accumulated by Migration Manager in the DMERROR table. This feature is useful to implementers because they can review the instances of bad data that affected package deployment. The error records for a particular package can be reviewed directly in the Migration Manager user interface or by running an Ad Hoc report against the DMERROR table for a particular package. By periodically reviewing accumulated data errors, implementers can avoid future packaging issues by identifying the correct set of system configurations and ensuring bad data hot spots are reduced or eliminated in the source environment.

Figure 15-39 shows a sample Ad Hoc report that was generated from the DMERROR table for a particular package.



Package Deployment Errors 2

Package Definition Name	Type	Migration Object	Keys	Message Details		Date
JHG_Linear_Cost	DDMETADATA	DMMAXOBJECTCFG	OBJECTNAME=WORKORDER;	psdi.util.MXApplicationException: BMXAA4191E - The value PLUSSSTATEPROVINCE is not valid for Domain. Specify a valid value for Domain. at psdi.app.configure.MaxAttributeCfg.throwNotValid(MaxAttributeCfg.java:2946) at psdi.app.configure.MaxAttributeCfg.validateDomain(MaxAttributeCfg.java:2349) at psdi.app.configure.MaxAttributeCfg.validateDomain(MaxAttributeCfg.java:2302) at psdi.app.configure.MaxAttributeCfg.appValidate(MaxAttributeCfg.java:939) at psdi.mbo.Mbo.validate(Mbo.java:4116) at psdi.mbo.MboSet.validate(MboSet.java:4925) at psdi.app.configure.MaxAttributeCfgSet.validate(MaxAttributeCfgSet.java:1545) at psdi.app.configure.MaxObjectCfg.appValidate(MaxObjectCfg.java:743) at psdi.mbo.Mbo.validate(Mbo.java:4116) at psdi.mbo.MboSet.validate(MboSet.java:4925) at psdi.mbo.MboSet.validateTransaction(MboSet.java:7301) at psdi.txn.MXTransactionImpl.validateTransaction(MXTransactionImpl.java:375) at psdi.txn.MXTransactionImpl.saveTransaction(MXTransactionImpl.java:207) at psdi.txn.MXTransactionImpl.save(MXTransactionImpl.java:156) at psdi.iface.mos.MosProcessImpl.save(MosProcessImpl.java:1757) at psdi.dm.procclass.DMMaxObjProcess.save(DMMaxObjProcess.java:668) at psdi.iface.mos.MosProcessImpl.processExternalData(MosProcessImpl.java:323) at psdi.iface.mos.MosProcessImpl.processObjectStructureService(MosProcessImpl.java:208) at psdi.iface.mic.MicService.processObjectStructureData(MicService.java:1908) at psdi.iface.mic.MicService.loadData(MicService.java:1693) at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1404) at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:713) at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:333) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25) at java.lang.reflect.Method.invoke(Method.java:597) at psdi.webclient.system.session.AsyncProcess.run(AsyncProcess.java:144) at java.lang.Thread.run(Thread.java:662)	3/13/12	4:09:10 PM

Figure 15-39 Sample Ad Hoc report

The following attributes are useful to display in an Ad Hoc report:

- Package
- Migration Object
- Keys
- Date
- Message
- Message Details

DMERROR table: The DMERROR table stores valuable data, including complete XML-document representations of the failing system configurations. The CLOB data columns that are storing XML can be added into the report. However, depending upon the size of the XML data, the report might be difficult to read. Better value is gained by leaving the CLOB data columns out of the report.

15.10.4 Product compatibility

In Chapter 2, “What is new in Migration Manager” on page 37, we introduced the product compatibility improvements that were made to Migration Manager as part of the Tivoli’s process automation engine release 7.5. In this section, we describe the key characteristics of the feature.

Migration Manager package deployments involve importing data into a target environment by using business objects and applying data validations to ensure that the data is clean. Often, Migration Manager is run in product environments where the set of products that are installed in a source are different from the set of products that are installed in a target. This difference implies that the data model for business applications, relationships among business objects, and the data validation rules are different between source and target. Migration Manager is cognizant of these differences and issues a warning to implementers that highlights the differences. It is the responsibility of the implementers to review, interpret, and decide whether migration is warranted under the circumstances. From our experience, migration of the platform system configurations can be run safely even when the product combinations are different between source and target.

From the Migration Manager perspective, product compatibility can be characterized by the following features:

- Checks performed by Migration Manager comparing the product information of source and target environment soon after deployment was initiated.
- User interface that is presented by Migration Manager to implementers that lists the products that are installed in source and target and highlighting the differences in products, fix packs, and test fixes.

In earlier releases of the Tivoli’s process automation engine, Migration Manager offered pop-up messages to implementers when differences were found in the product combinations. While informative, these messages were not user-friendly. In addition, the Migration Manager did not allow deployment of package when the source environment contained more products that the target environment did not include. This restrictive implementation approach was chosen because it was anticipated that customers had similar development and production environments.

Figure 15-40 shows the product compatibility approach of Migration Manager in older releases of Tivoli's process automation engine.

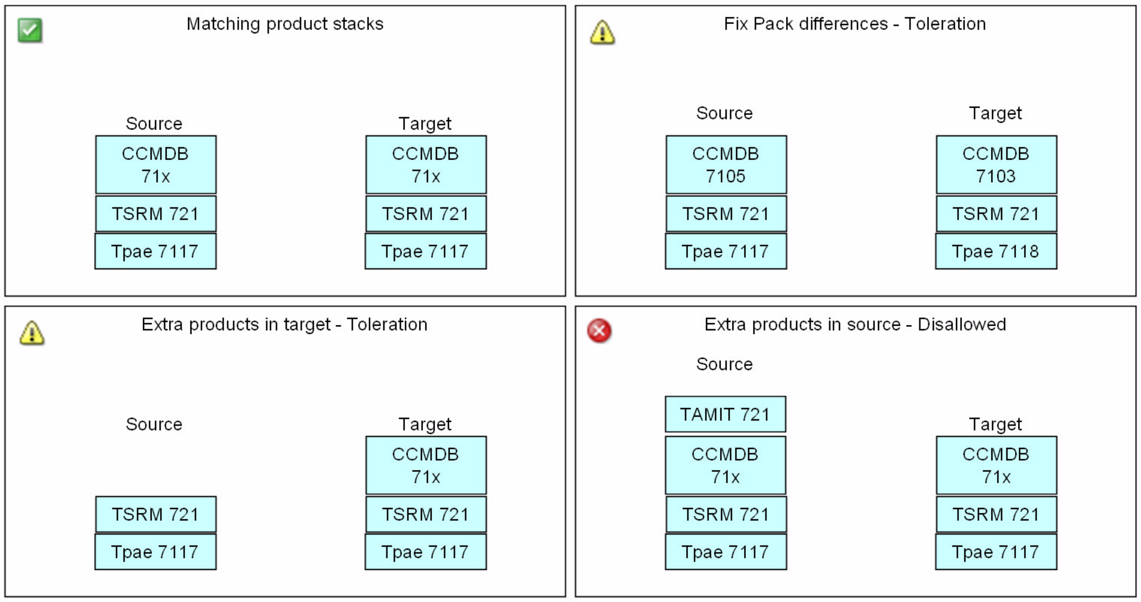


Figure 15-40 Migration Manager product compatibility approach in Tivoli's process automation engine 7.1

In the chart, the upper left quadrant shows the recommended model of maintaining the same product stacks for source and target. Because this model might not always be practical, the upper right and lower left quadrants show Migration Manager's toleration of fix pack and extra products in target. The lower right quadrant shows the path that Migration Manager did not permit implementers to use.

However, this approach did not take into account the fact that many products and add-ons contained merely best practices content and application data and no code. We also observed that many customer implementations, while allowing installation of multiple products and add-ons in a development environment, restricted what products were installed into a production environment. In response to the ground reality with product packaging and installation scenarios, this restriction was removed with the Tivoli's process automation engine 7.5 release.

Figure 15-41 shows the Migration Manager tolerating extra products in target.

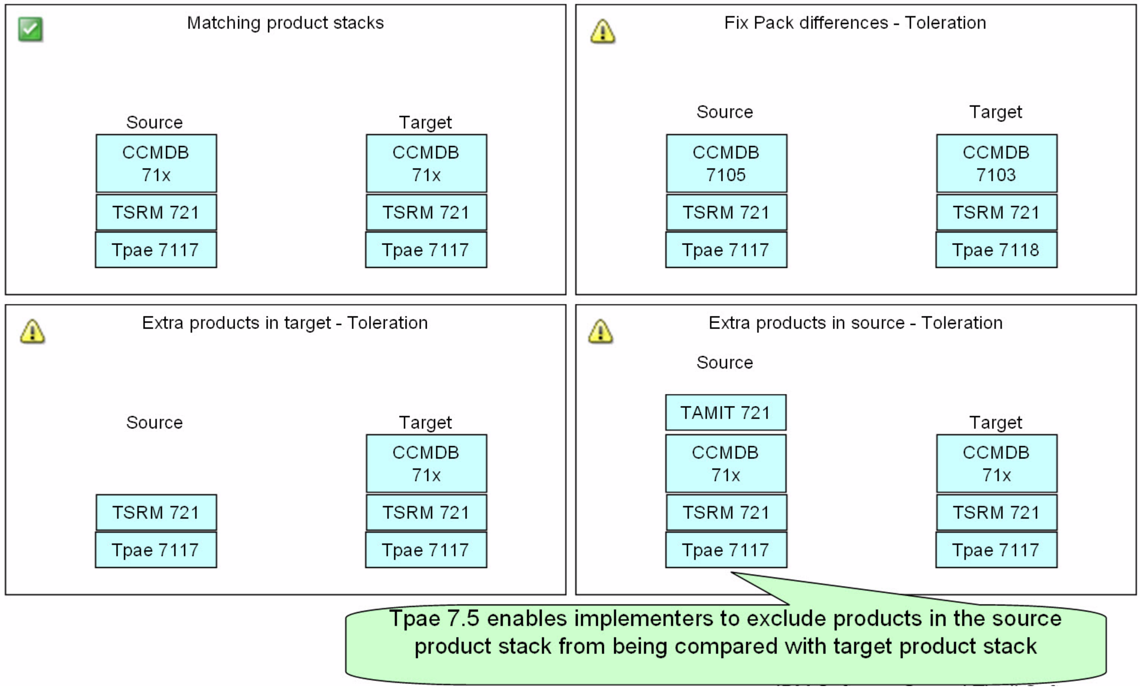


Figure 15-41 Migration Manager product compatibility approach for Tivoli's process automation engine 7.5

All product compatibility information is presented by Migration Manager in the Source and Target Product Compatibility window. This window is presented at the time of deployment if Migration Manager finds at least one difference. Table 15-9 lists the messages that are presented in the window for an implementer to review.

Table 15-9 Product compatibility messages

Product Compatibility Message	Remarks
One or more extra products are in the source.	Implementer must check whether the system configurations to be deployed into target are not specific to a product that is present only in the source environment.
The modification level or fix pack level does not match.	Implementer must check whether the system configurations include a dependency or use modification level or fix pack level metadata or functionality.
The version level or release level does not match.	Migration Manager does not allow deployment under these conditions. For, example, a Change and Configuration Management Database 7.1 migration package cannot be deployed into SCCD 7.5.
The product and all its version levels match.	Product environments match, which is the recommended state for participating product environments.

15.11 Setting up logging for the Migration Manager

The following capabilities of the Migration Manager and Migration Collections might periodically require detailed product logs to isolate and resolve errors:

- Migration Manager tasks that pertain to package management
- Migration Manager comparisons
- Migration Collection validation

In this section, we describe the log configuration and offer corresponding sample log outputs for these capabilities. Understanding the nature of the log statements can help implementers process log information more rapidly and interpret the behavior and isolate underlying issues.

For more information about the general use of the logging application, see these resources:

- *Migration Manager Guide* at this website:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tamit.doc_7.1/pdf/mam71_migration_mgr_guide.pdf

- *IBM Maximo Asset Management, IBM Tivoli Asset Management for IT, and IBM Tivoli Service Request Manager System Administrator Guide* at this website:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tamit.doc_7.1/pdf/mam71_sys_admin_guide.pdf

In this section, we describe specific ways to set up the system to provide the necessary output that is required to troubleshoot any failed migration packages.

Complete the following steps to set up logging to provide optimal results:

1. Set up the root logger.
2. Set up the root logging folder.
3. Set up an appender for logging the Migration Manager output.
4. Set the logging level for the Migration Manager service.

Manage the root logger

Complete the following steps to manage the root logger:

1. Browse to the Logging application. Click **Select Action**, as shown in Figure 15-42.

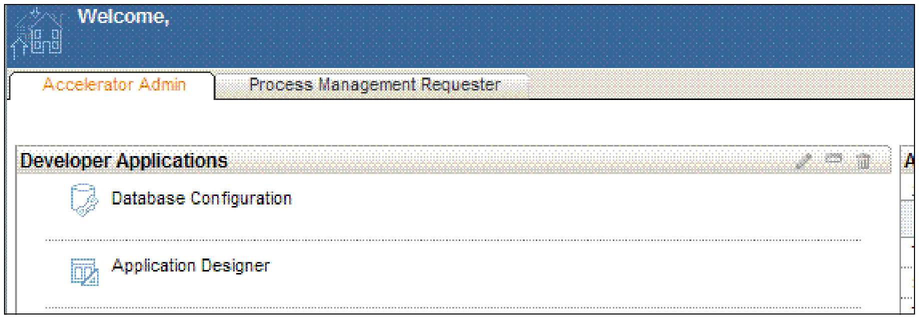
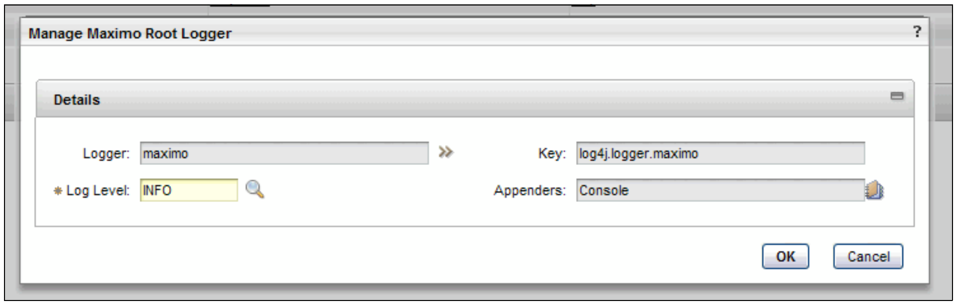


Figure 15-42 Manage the root logger

2. Click **Manage Maximo Root Logger**. A new panel appears, as shown in Figure 15-43.


Manage Maximo Root Logger

Details

Logger: maximo

Key: log4j.logger.maximo

Log Level: INFO

Appenders: Console

Figure 15-43 The root logger panel

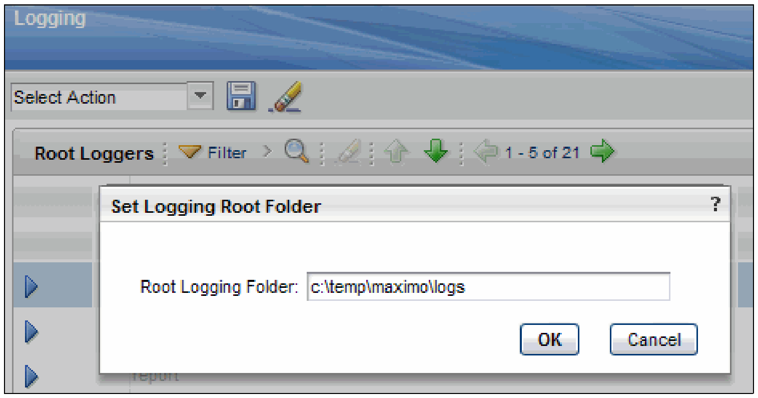
3. You can set the root logging level and change the main appender. If nothing specific is set for any one service, all of the logging for the system is affected.

Setting the logging folder

Next, you must set up the root logging folder. If you decide not to set up the root logging folder, be aware that the location of logs for IBM WebSphere®, by default, without setting the root folder is normally in the following folder structure:

```
\\ibm\WebSphere\AppServer\profiles\ctgAppSrv01\maximo\logs
```

Figure 15-44 shows how to set the Root logging folder if you want to set the root folder to another location. This window is accessed by clicking **Select Action** → **Set Logging Root Folder**.


Set Logging Root Folder

Root Logging Folder: c:\temp\maximo\logs

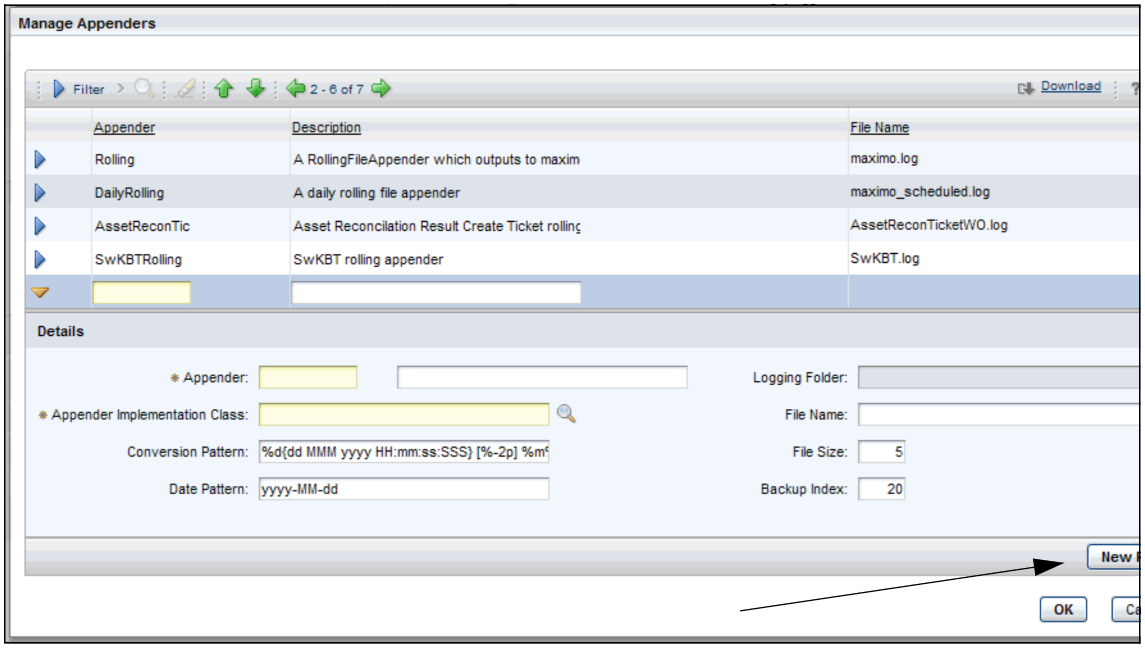
Figure 15-44 A custom root logging folder definition

Creating an appender

The next step is to create a unique appender for you to isolate the Migration Manager activity from all other logging activities. By using this task, you can more easily read the error log when you must isolate an error incident.

Complete the following steps to create an appender:

1. Browse to the Manage Appenders window, as shown in Figure 15-45, by clicking **Select Action** → **Manage Appenders**.



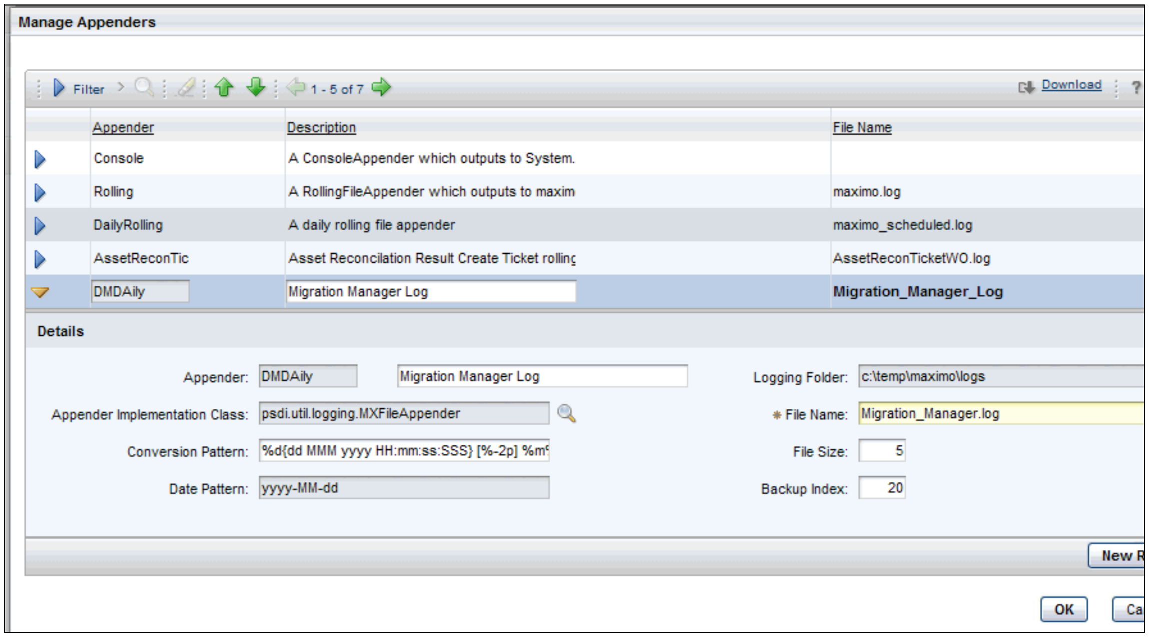
Appender	Description	File Name
Rolling	A RollingFileAppender which outputs to maxim	maximo.log
DailyRolling	A daily rolling file appender	maximo_scheduled.log
AssetReconTic	Asset Reconcilation Result Create Ticket rolling	AssetReconTicketWO.log
SwKBTRolling	SwKBT rolling appender	SwKBT.log

* Appender:	
* Appender Implementation Class:	
Conversion Pattern:	%d{dd MMM yyyy HH:mm:ss:SSS} [%-2p] %m
Date Pattern:	yyyy-MM-dd
Logging Folder:	
File Name:	
File Size:	5
Backup Index:	20

Figure 15-45 Manage Appenders window

2. Click **New Row** and enter the name of the appender that you want to create. Add the description, and the, select the **Appender Implementation Class**.

3. Add the File Name that you want the appender to use, and then click **OK**, as shown in Figure 15-46 on page 413.


Manage Appenders

Filter 1 - 5 of 7 Download

	Appender	Description	File Name
▶	Console	A ConsoleAppender which outputs to System.	
▶	Rolling	A RollingFileAppender which outputs to maxim	maximo.log
▶	DailyRolling	A daily rolling file appender	maximo_scheduled.log
▶	AssetReconTic	Asset Reconcilation Result Create Ticket rolling	AssetReconTicketWO.log
▼	DMDAily	Migration Manager Log	Migration_Manager_Log

Details

Appender:	DMDAily Migration Manager Log	Logging Folder:	c:\temp\maximo\logs
Appender Implementation Class:	psdi.util.logging.MXFileAppender	* File Name:	Migration_Manager.log
Conversion Pattern:	%d{dd MMM yyyy HH:mm:ss:SSS} [%-2p] %m	File Size:	5
Date Pattern:	yyyy-MM-dd	Backup Index:	20

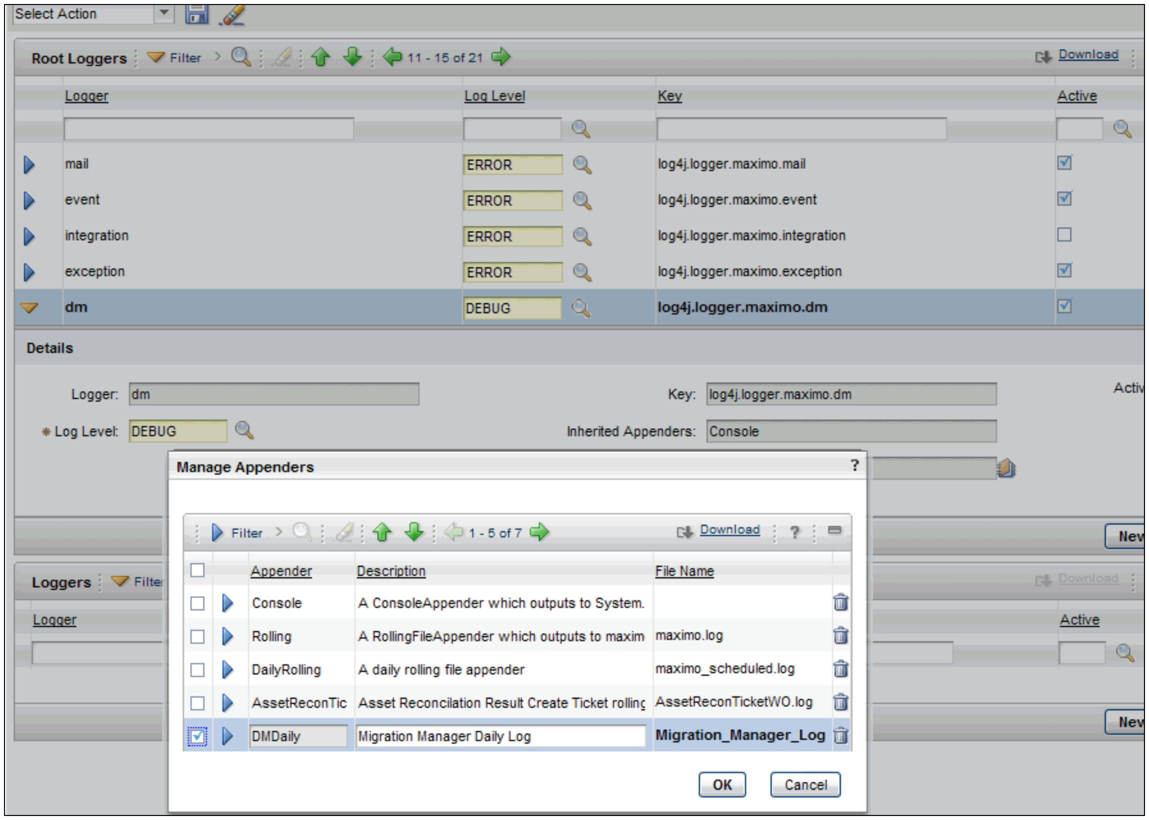
Figure 15-46 Example of a new appender

You also can set the file size and the backup index. We recommend the use of the default settings until you experience demonstrates a need for change.

Setting the logging level for the Migration Manager service

Complete the following steps to access the specific root logger for the Migration Manager in the list search for dm:

1. Click the arrow to expand the dm entry, as shown in Figure 15-47. Set the Log Level to `DEBUG`.



Root Loggers — 11 - 15 of 21

Logger	Log Level	Key	Active
mail	ERROR	log4j.logger.maximo.mail	☑
event	ERROR	log4j.logger.maximo.event	☑
integration	ERROR	log4j.logger.maximo.integration	☐
exception	ERROR	log4j.logger.maximo.exception	☑
dm	DEBUG	log4j.logger.maximo.dm	☑

Details

Logger: dm

Key: log4j.logger.maximo.dm

Log Level: DEBUG

Inherited Appenders: Console

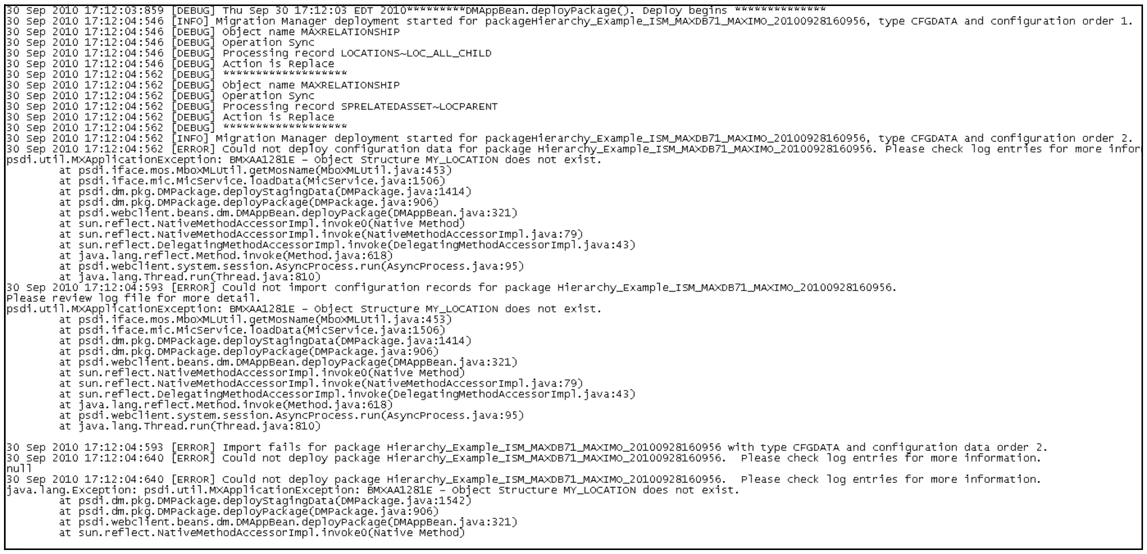
Manage Appenders — 1 - 5 of 7

	Appender	Description	File Name
☐	Console	A ConsoleAppender which outputs to System.	
☐	Rolling	A RollingFileAppender which outputs to maxim	maximo.log
☐	DailyRolling	A daily rolling file appender	maximo_scheduled.log
☐	AssetReconTic	Asset Reconcilation Result Create Ticket rolling	AssetReconTicketWO.log
☑	DMDaily	Migration Manager Daily Log	Migration_Manager_Log

Figure 15-47 Setting the appender

2. Click the icon next to the Appender field and select the appender that you created. Click the small square before the appender DWDaily. Click **OK**.
3. Click **Save**, then click **Select Action** → **Apply Settings**.
4. Accept the dialog box responses and then use the system.

Figure 15-48 shows the result of this process after a package is processed with an error. For more information, see Figure 16-5 on page 435.



```
30 Sep 2010 17:12:03:859 [DEBUG] Thu Sep 30 17:12:03 EDT 2010*********DMAppBean.deployPackage(). Deploy begins **************
30 Sep 2010 17:12:04:546 [INFO] Migration Manager deployment started for packageHierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956, type CFGDATA and configuration order 1.
30 Sep 2010 17:12:04:546 [DEBUG] Object name MAXRELATIONSHIP
30 Sep 2010 17:12:04:546 [DEBUG] Operation Sync
30 Sep 2010 17:12:04:546 [DEBUG] Processing record LOCATIONS~LOC_ALL_CHILD
30 Sep 2010 17:12:04:546 [DEBUG] Action is Replace
30 Sep 2010 17:12:04:562 [DEBUG] *******************
30 Sep 2010 17:12:04:562 [DEBUG] Object name MAXRELATIONSHIP
30 Sep 2010 17:12:04:562 [DEBUG] Operation Sync
30 Sep 2010 17:12:04:562 [DEBUG] Processing record SPRELATEDASSET~LOCPARENT
30 Sep 2010 17:12:04:562 [DEBUG] Action is Replace
30 Sep 2010 17:12:04:562 [DEBUG] *******************
30 Sep 2010 17:12:04:562 [INFO] Migration Manager deployment started for packageHierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956, type CFGDATA and configuration order 2.
30 Sep 2010 17:12:04:562 [ERROR] Could not deploy configuration data for package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956. Please check log entries for more infor
psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
        at psdi.iface.mos.MboXMLUtil.getMosName(MboXMLUtil.java:453)
        at psdi.iface.mic.MicService.loadData(MicService.java:1506)
        at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1414)
        at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:906)
        at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:321)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:79)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:618)
        at psdi.webclient.system.session.AsyncProcess.run(AsyncProcess.java:95)
        at java.lang.Thread.run(Thread.java:810)
30 Sep 2010 17:12:04:593 [ERROR] Could not import configuration records for package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956.
Please review log file for more detail.
psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
        at psdi.iface.mos.MboXMLUtil.getMosName(MboXMLUtil.java:453)
        at psdi.iface.mic.MicService.loadData(MicService.java:1506)
        at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1414)
        at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:906)
        at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:321)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:79)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:618)
        at psdi.webclient.system.session.AsyncProcess.run(AsyncProcess.java:95)
        at java.lang.Thread.run(Thread.java:810)

30 Sep 2010 17:12:04:593 [ERROR] Import fails for package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956 with type CFGDATA and configuration data order 2.
30 Sep 2010 17:12:04:640 [ERROR] Could not deploy package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956.  Please check log entries for more information.
null
30 Sep 2010 17:12:04:640 [ERROR] Could not deploy package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928160956.  Please check log entries for more information.
java.lang.Exception: psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
        at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1542)
        at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:906)
        at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:321)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

Figure 15-48 Sample error log

15.12 Start Center visibility into configurations

It is beneficial to managers, developers, and implementers to view the key configurations that are set up in a development environment; not only from within a particular application, such as Workflow Designer, but also from a Start Center. This at-a-glance Start Center-based view into key configurations enables stakeholders to determine the number and extent of configurations and prepare Snapshot migration packages.

A Start Center can be designed rapidly to deliver this type of a view, if a consistent naming convention was used to create the various configurations. Queries can be constructed to retrieve the wanted configurations that are based on the naming convention that was used. Start Center result set portlets can be associated with those queries. Each portlet can be configured to display the most relevant information from the development and migration perspectives.

For example, to retrieve and display workflow processes that might be required to be migrated, the following simple query can work:

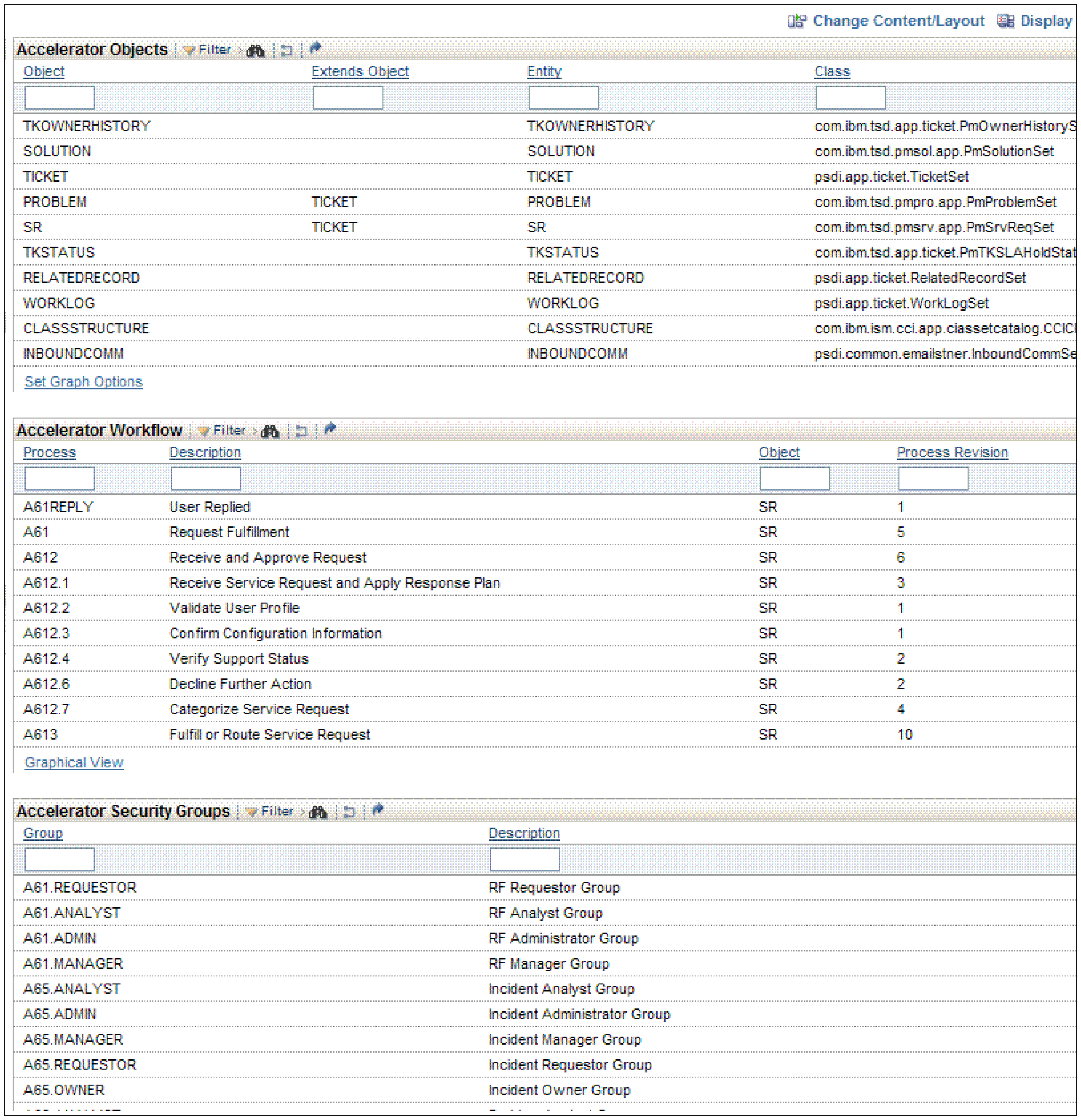
```
processname like 'RB%' and active = '1'
```

This query now can be associated with a Start Center result set portlet that displays the following key workflow process information:

- Process
- Description
- Object
- Process revision
- Enabled
- Active
- Changed by
- Changed date

When the Start Center is saved and associated with the appropriate security groups, the key information is immediately visible to users when they log in to the production environment. Figure 15-49 on page 417 shows an example Start Center (only part of the window is shown) on which portlets provide visibility into business objects, workflow processes, and security groups.

Important: With the Tivoli's process automation engine 7.5 release, we believe that Migration Collections serves as the centralized location to manage system configurations that are packaged with Migration Manager. While the Start Center can be used to display system configurations, the better approach is to use Migration Collections directly to view, collect, and manage system configurations for migration.



Change Content/Layout Display

Accelerator Objects Filter

Object	Extends Object	Entity	Class
TKOWNERHISTORY		TKOWNERHISTORY	com.ibm.tsd.app.ticket.PmOwnerHistoryS
SOLUTION		SOLUTION	com.ibm.tsd.pmsol.app.PmSolutionSet
TICKET		TICKET	psdi.app.ticket.TicketSet
PROBLEM	TICKET	PROBLEM	com.ibm.tsd.pmpro.app.PmProblemSet
SR	TICKET	SR	com.ibm.tsd.pmsrv.app.PmSrvReqSet
TKSTATUS		TKSTATUS	com.ibm.tsd.app.ticket.PmTKSLAHoldStat
RELATEDRECORD		RELATEDRECORD	psdi.app.ticket.RelatedRecordSet
WORKLOG		WORKLOG	psdi.app.ticket.WorkLogSet
CLASSSTRUCTURE		CLASSSTRUCTURE	com.ibm.ism.cci.app.classetcatalog.CCICl
INBOUNDCOMM		INBOUNDCOMM	psdi.common.emailstner.InboundCommSe

Set Graph Options

Accelerator Workflow Filter

Process	Description	Object	Process Revision
A61REPLY	User Replied	SR	1
A61	Request Fulfillment	SR	5
A612	Receive and Approve Request	SR	6
A612.1	Receive Service Request and Apply Response Plan	SR	3
A612.2	Validate User Profile	SR	1
A612.3	Confirm Configuration Information	SR	1
A612.4	Verify Support Status	SR	2
A612.6	Decline Further Action	SR	2
A612.7	Categorize Service Request	SR	4
A613	Fulfill or Route Service Request	SR	10

Graphical View

Accelerator Security Groups Filter

Group	Description
A61.REQUESTOR	RF Requestor Group
A61.ANALYST	RF Analyst Group
A61.ADMIN	RF Administrator Group
A61.MANAGER	RF Manager Group
A65.ANALYST	Incident Analyst Group
A65.ADMIN	Incident Administrator Group
A65.MANAGER	Incident Manager Group
A65.REQUESTOR	Incident Requestor Group
A65.OWNER	Incident Owner Group

Figure 15-49 Sample Start Center that shows business objects, workflow processes, and security groups

16

Troubleshooting

This chapter describes the approach for the implementer to use when the migration packages do not migrate as expected. We present the most common situations we experienced. We provide detailed techniques to help you determine the cause of the migration failure. We also offer several techniques that can help prevent or recover rapidly from migration failures.

This chapter includes the following sections:

- Common migration failures
- Methods to solve migration package failures
- Techniques to prevent migration package failures

16.1 Common migration failures

This section describes the most common occurrences for migration failures that we encountered. The troubleshooting section of the *Migration Manager Guide* describes the three major types of errors that are found when the application is used: package creation, distribution, and deployment. This section describes these failures.

For more information about the troubleshooting section of the Migration Manager Guide, see this website:

`http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/gp_migmgr/c_migration_mgr_msg.html`

This section is not intended to be an exhaustive list of all potential Migration Manager failures. In addition to listing common Migration Manager errors, product documentation offers a list of Migration Manager limitations. The documentation limitations should be reviewed with the information that is presented in this chapter. For more information about limitations, see this website:

`http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/gp_migmgr/c_mig_limitations.html`

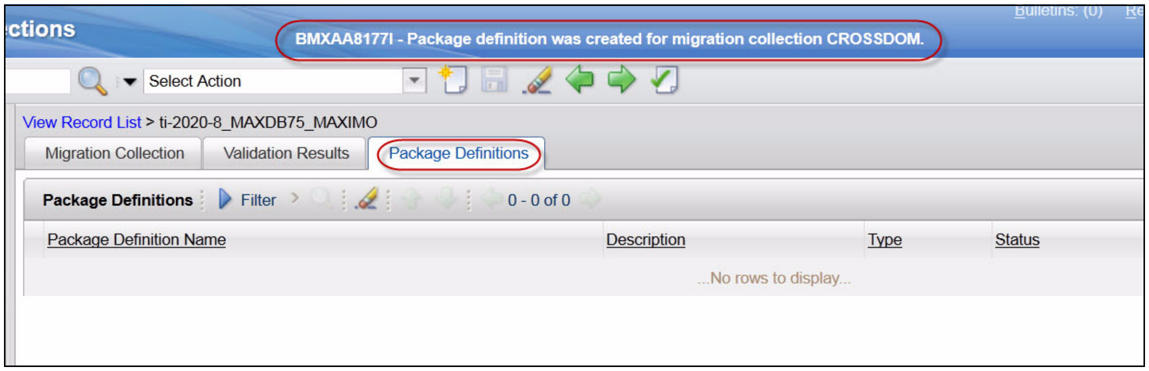
16.1.1 Package creation from migration collection fails

Under certain conditions, creation of a package from a migration collection might not work as expected. We explain the root cause for this issue and describe how the issue can be easily overcome. This issue is not so much a failure as behavior by design that must be well-understood by an implementer.

The Migration Collections application supports scenarios where system configurations have circular dependencies. For example, system configurations can be defined such that a crossover domain depends upon an object whose attribute in turn references the same crossover domain. Such a circular dependency is dealt with by separating the object system configuration from the crossover domain system configuration and packaging each separately.

Migration Collections application recognizes these situations by applying Package Definition Exceptions. Package Definition Exceptions are rules that are applied to the contents of a collection to determine whether circular dependencies exist and then generate separate package definitions for the system configurations that cause such dependencies.

When the Create Package Definition action is run from the Select Action menu of the Migration Collections application, collections application determines whether exceptions must be applied and the system configurations that are split across package definitions. For the crossover domain and object scenario, the crossover domain and business object must be included in the collection for the situation to be recognized and the system configurations to be successfully split. If one of the system configurations that is causing circular dependency is not present in the collection, package creation fails. Although a confirmation message is displayed in package creation, the Package Definitions tab remains empty, as shown in Figure 16-1.


BMXAA8177I - Package definition was created for migration collection CROSSDOM.

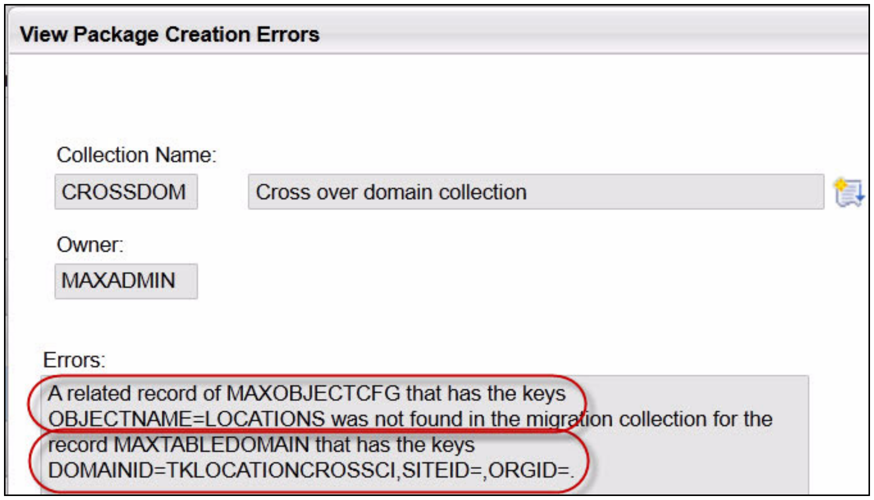
View Record List > ti-2020-8_MAXDB75_MAXIMO

Package Definition Name	Description	Type	Status
...No rows to display...			

Figure 16-1 Results of package definition creation with circular system configurations

If this situation is encountered, you should run the View Package Creation Errors action. A pop-up window displays the list of circular dependencies and indicates which particular dependency is missing from the collection. In the crossover domain example, the error message indicates that the crossover domain TKLOCATIONCROSSCI depends upon the business object LOCATIONS.

Figure 16-2 shows the error message that is displayed and indicates which two system configurations must be present in the collection.


View Package Creation Errors

Collection Name: CROSSDOM Cross over domain collection

Owner: MAXADMIN

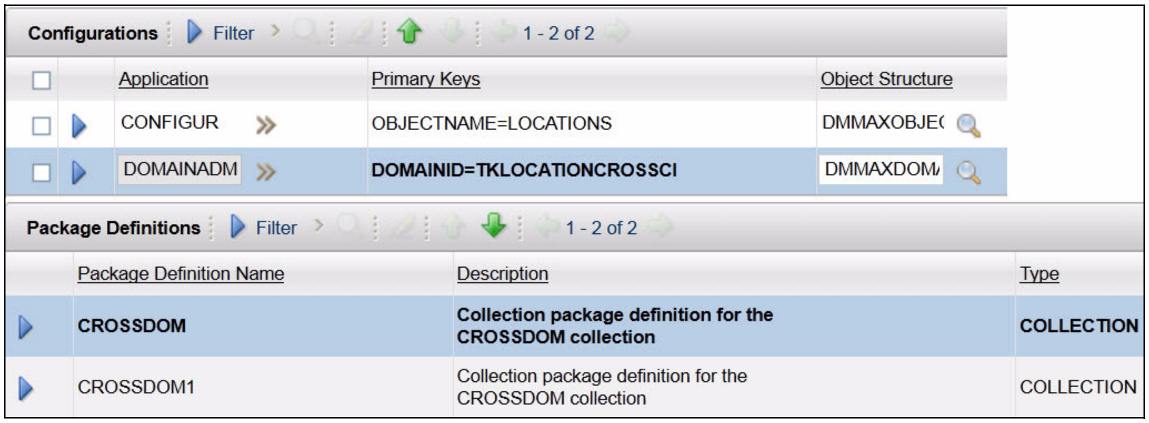
Errors:
A related record of MAXOBJECTCFG that has the keys OBJECTNAME=LOCATIONS was not found in the migration collection for the record MAXTABLEDOMAIN that has the keys DOMAINID=TKLOCATIONCROSSCI,SITEID=,ORGID=.

Figure 16-2 Package definition creation error that indicates missing system configuration

Important: In Figure 16-2, the View Package Creation Errors pop-up window displays a single message. However, depending upon the number of collection entries that are identified as circular dependencies, more than one error message might be displayed. Review each message and ensure both configurations that are identified in the message are included in the collection.

Package creation errors such as these can be resolved by adding the dependent system configuration into the collection and running the Create Package Definition action. For the crossover domain scenario, adding the LOCATIONS object into the collection resolves the package definition creation error.

Figure 16-3 shows the collection that contains both object and domain and the resulting generation of the package definition.



	Application	Primary Keys	Object Structure
☐	CONFIGUR	OBJECTNAME=LOCATIONS	DMMAXOBJE(
☐	DOMAINADM	DOMAINID=TKLOCATIONCROSSCI	DMMAXDOM/

Package Definition Name	Description	Type
CROSSDOM	Collection package definition for the CROSSDOM collection	COLLECTION
CROSSDOM1	Collection package definition for the CROSSDOM collection	COLLECTION

Figure 16-3 System configurations with circular dependencies are successfully split

The key to resolving a package creation error issue with collections is to review and understand package exception rules, recognize when circular dependencies must be migrated, and ensure that instructions that are associated with each package exception rule are followed.

16.1.2 Package creation fails in source environment

In the experience of the authors, package creation failures are rare. However, they do occur under certain circumstances. This section describes two scenarios in which package creation fails in Migration Manager.

During the creation of a package, system configuration data is extracted from the underlying database tables and converted into an XML document. Upon successful creation, the XML document is stored in the DMPKGSTAGING table of the Tivoli’s process automation engine database. Under specific circumstances, the creation of the XML document fails. We describe two different manifestations of the inability to successfully create the XML document.

Invalid domain value scenario

In this manifestation, package creation fails with a domain validation error. The error might be similar to the following examples:

- `BMXAA5324E - Could not extract or write xml document for staging table record for migration object DMMAXUSER for package <package name> . BMXAA4024E - The synonym value PRIMARY is not valid for the domain USERTYPE.`
- `BMXAA5324E - Could not extract or write xml document for staging table record for migration object DMACTION for package <package name> . BMXAA4024E - The synonym value CREATEPO is not valid for the domain APPACTION.`

Data that is extracted during package creation might contain values that were declared as synonym domain valid values. As part of data extraction, the internal value of a synonym domain value might be retrieved by the underlying business object that is based on the external value that is specified in the system configuration. If the external value is invalid and the corresponding internal value cannot be retrieved, an exception is thrown, which results in stopping the data extraction with error BMXAA5324E.

In the second example that is provided, data extraction fails because an action system configuration references an external value CREATEPO, which does not exist in the APPACTION synonym domain.

This failure is the result of bad synonym domain data in the system configuration that is targeted for migration. The data must be corrected before a package can be created with the affected system configuration. Running SQL statements against the affected system configuration and the corresponding synonym domain can help identify data discrepancies. For example, to resolve the APPACTION error, the following SQL can be run:

```
select * from action where type='APPACTION' and value2 not in (select
value from synonymdomain where domainid='APPACTION')
```

If running SQL results in zero records, there is a match between values that are specified in a system configuration and values that are declared in the corresponding synonym domain. But if one or more records are returned, the corresponding system configuration records in ACTION table should be reviewed. A decision should be made if records in ACTION table must be updated or the synonym domain must be updated with a new synonym value.

Similar SQL statements might have to be run against other system configuration tables if the BMXAA4024E error is reported against business objects.

With this publication, we ship a file with a set of SQL commands implementers that are compiled to assist with this type of failure. For more information and download instructions, see Appendix A, “Additional material” on page 445.

Invalid XML characters scenario

This scenario is applicable only to product releases that run Tivoli's process automation engine 7.1.1.6 or lower.

In this scenario, package creation fails with an XML invalid characters error. The error might be similar to the following example:

```
BMXAA5324E - Could not extract or write xml document
to staging table record for migration object DMWFPROCESS for package
<package name>.BMXAA5822E - The MBO WFINTERACTION [with unique id 510]
field DIRECTIONS_LONGDESCRIPTION has XML 1.0 invalid characters.
```

The resolution to this scenario is to apply the latest 7.1.x fix pack for the Tivoli's process automation engine. Higher releases of the fix pack ensure that the text in the columns that are exported are correctly binary encoded during the extraction into XML document.

16.1.3 Package deployment fails because of installation differences

With the Tivoli's process automation engine release 7.5, improvements were made to accommodate differences between the product versions and releases that are used in a source environment as compared with a target environment. We described Migration Manager’s product compatibility improvements in Chapter 15, “Common topics” on page 351. This improvement is intended to relax earlier restrictions that the Migration Manager applied to migrations. It does not imply that any data can be successfully migrated between disparate environments.

Best practice: Implementers should strive to maintain product environments with the same set of products and fix packs.

We encountered situations where migrations fail because of product installation differences. In some situations, package deployment might be successful, but the system configurations might cause unexpected errors in business applications and therefore become harder to isolate and resolve.

We offer an example of this situation in this section.

In a development environment that implements Maximo and Scheduler, custom business objects and associated Java code are defined. A package of these custom business objects is prepared. The object definition includes references to the Java code that was authored in support of the objects. This package is to be deployed into a test environment that implements Maximo and the Scheduler add on. A Scheduler test fix was applied into the target environment. After this test fix was applied, packages with custom objects were deployed without error. However, when testing, it is determined that the functionality that is associated with Preventive Maintenance application is now broken. If the Generate Work Orders action is run from the Preventive Maintenance application, an error message is displayed on the application user interface that indicates an unknown error occurred.

Upon investigating the issue, the root cause was found to be a Java ClassCastException in the target environment that results from the Scheduler product code that cannot run as expected. The custom objects that were migrated earlier registered Java classes that triggered the ClassCastException.

Important: Further investigation revealed that the Scheduler test fix should be applied only after the migration packages were deployed and not before. The resolution for this situation was to back out the test fix, deploy the packages, and reapply the test fix.

Such costly situations can be avoided by consciously reviewing and comparing products, releases, and fix packs that are installed in sources and targets and assessing if migration packages affect target environments.

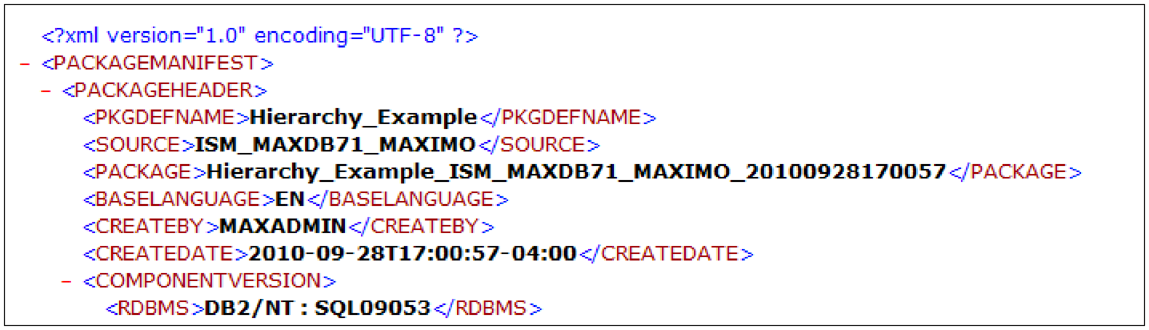
Best practice: System Information from the participating environments should be retrieved through the user interface of the products and reviewed. If different products and Fix Packs are identified to be installed, an assessment should be done of the potential impact and order of migration.

16.1.4 Package deployment fails because of incorrect Source designation

Deployments fail if you choose a package incorrectly. This situation occurs when you use a central file server to store packages for several environments. This situation is not that uncommon and occurs most often when you develop a project and have many packages that are all in the same folder.

Review the package manifest

The package manifest provides you with the information that is needed to analyze the contents of the package. Often, the package manifest reveals that the Source system is the consuming environment. By using the manifest, you can deploy the package correctly. In Figure 16-4, you see the Source system information.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <PACKAGEMANIFEST>
  - <PACKAGEHEADER>
      <PKGDEFNAME>Hierarchy_Example</PKGDEFNAME>
      <SOURCE>ISM_MAXDB71_MAXIMO</SOURCE>
      <PACKAGE>Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057</PACKAGE>
      <BASELANGUAGE>EN</BASELANGUAGE>
      <CREATEBY>MAXADMIN</CREATEBY>
      <CREATEDATE>2010-09-28T17:00:57-04:00</CREATEDATE>
    - <COMPONENTVERSION>
        <RDBMS>DB2/NT : SQL09053</RDBMS>
```

Figure 16-4 Package manifest XML

The XML tag Source contains the information that you must validate against your Target system.

16.1.5 Package deployment fails because of inconsistent manifest data with package file name

When you download the migration package file from your Source server to your computer, the file name might change unexpectedly. This situation creates an inconsistency that causes the deployment to fail.

The name of the file changes when the browser settings rename the downloaded file if the browser sees the file already in its downloaded location. This situation happens most often with metadata.

To determine if so, browse to the download location and make a note of the file name of the package. Open the package and extract the manifest. If the name of the package in the manifest does not match the name of the compressed file, rename the package compressed file to match the information in the manifest.

16.1.6 Package deployment fails because of improperly combined object structures

The following error occurs when you associate one migration group to another migration group on which there is a dependency and you do not use the dependency. This error occurs because dependencies are processed and committed into the database before the next group that depends on that data.

Herein lies a challenge for you. If your dependent data group is large, but it depends on a small part of a large data migration group, setting up the dependency in the migration group or package affects performance. This situation is true because there is no way of filtering the dependent data group. Therefore, the following choices are available to alleviate this problem:

- You can create a migration package with only the particular object structure in question. Then, filter the package for the specific small record set. Migrate this package, and then migrate the original package without the dependency. Because you just processed the data on which the original set is dependent, you do not encounter problems.
- You can create the dependency and process the data. This method takes longer and, depending upon the amount of data in the dependency, it might take even longer to process the migration.

16.1.7 Security configuration migration errors

In this section, we cover the migration errors that are related to the security configuration.

Security group and Start Center association

Every security group is associated with a corresponding Start Center. If a user belongs to multiple security groups, that user might be assigned to multiple Start Centers as a result of the accumulation of Start Centers that are associated with each security group.

When a security group is migrated, be careful to ensure that the associated Start Center was migrated in a separate package or precedes the security group configuration in the current package. If this task is not done, the migration of the security group fails with the following error:

```
[ERROR] java.lang.NullPointerException at
psdi.dm.procclass.DMMaxGroupProcess.setAdditionalData
(DMMaxGroupProcess.java:114)
```

During the deployment of the migration package that contains the security group, the Migration Manager attempts to establish the link between the security group and its associated Start Center. Because the Start Center does not exist in the Target production environment, the Migration Manager reports the deployment error.

An alternative approach to the security group migration is to exclude the reference to the Start Center before the migration is done. Complete the following steps to set this exclusion:

1. Access the Object Structure application and display the DMMAXGROUP object structure.
2. From the Select Action menu, run the Exclude/Include Fields action.
3. In the pop-up window that is shown, exclude the SCTEMPLATEID attribute from the MAXGROUP business object.
4. Click **OK** to save the change, and close the dialog.

This process ensures that the Start Center template reference is not carried into the migration package in the source environment. After the migration completes, you must manually establish the link between the security groups and the corresponding Start Centers. Consider this important trade-off part of the migration planning activities.

16.1.8 Security group and group reassign errors

When migrating users, the following deployment error is reported:

```
BMXAA6695E - The MBO could not be batch validated for object
GROUPUSER. The error is GROUPNAME
psdi.util.MXApplicationException: BMXAA0028E - Your security privileges
do not allow access to the selected option
```

Users are migrated with the DMMAXUSER object structure. Before users are migrated, security groups are migrated with the DMMAXGROUP object structure. In the DMMAXUSER object structure, the GROUPUSER business object associates users with the security groups of which they are members.

Specific security authorizations are required to assign a user to a group. A user is granted the authorization to add other users to a chosen group.

When migrating users with the Migration Manager, the user or administrator that is performing the deployment of the migration package must have the authorization to assign users to a security group. If this authorization is not granted, the validation rules against GROUPUSER Maximo Business Object (MBO) fail with the deployment error stated previously. Ensure that the user who is performing the migration is an administrative user that is granted requisite authorizations. Typically, to perform migrations, clients use the MAXADMIN user account or an account that is duplicated from MAXADMIN.

16.1.9 Impact of invalid or deprecated MAXVARS

Clients encountered deployment errors when the migration package included the DMMAXVARS object structure and the data for the MAXVARS table. The following error was reported:

```
BMXAA4116E - Maxvar type is not valid
```

The MAXVARS table holds a number of system-wide, organization-level, or site-level flags. Several of these flags can be set through various Select Action menus of the Organization application. Every MAXVARS table entry must be associated with a corresponding MAXVARTYPE table entry. The entry in the MAXVARTYPE table determines whether the particular MAXVAR is system-wide, an organization level, or site level. The MAXVAR business object enforces validations that ensure that the entries in the MAXVARS and MAXVARTYPE tables match.

MAXVARS and MAXVARTYPE entries are created solely through installation or upgrade programs. If the installation or upgrade of the same product completes in a production environment, the MAXVARs must exactly match between the two production environments. The MAXVAR business object does not offer the capability to add a new MAXVAR entry into the MAXVARS table.

The Migration Manager supports only the update of existing MAXVAR entries that are subject to the validations that are enforced by the MAXVAR business object.

Over the course of several product releases, a subset of MAXVAR is deprecated (no longer in use with the product).

IBM recognized a defect in the upgrade scripts that are supplied with Tivoli's process automation engine that causes orphan or deprecated MAXVAR entries to be left behind in the production environment after it is upgraded to release 7.1.*x*. The migration of MAXVAR that includes these entries causes the migration to fail. The defect was remedied with the release of Maximo Base Services (MBS) Fix Pack 7.1.1.6. For more information, see this website:

http://www-01.ibm.com/support/docview.wss?uid=swg1IZ61692

A preventive measure that clients can take is to run the following SQL statement in both environments to determine that the number of MAXVARS entries matches:

```
select count(*) from maxvars
```

The following SQL statement ensures that there is at least one corresponding entry in the MAXVARTYPE table for an entry in the MAXVARS table:

```
select count(*) from maxvars where exists (select * from maxvartype
where maxvars.varname=maxvartype.varname)
```

The number of records that are returned by the two SQL statements must match.

Another consideration to successfully migrate MAXVARS is to track the changes that were made in the development environment and migrate only those MAXVARs that were modified.

16.1.10 Attached documents and document types

When a Snapshot package is migrated with the Replace processing action, a client encountered the following deployment error:

```
Error occurred while processing DMDOCTYPES (Object Structure number 1.
Primary Object is: DOCTYPES. Key is: Attachments). Error is: BMXAA0876E
- Cannot delete this record
```

The DMDOCTYPES object structure supports the migration of attached document types. The DOCINFO business object is a member of this object structure and the child of the DOCTYPES business object. When a Snapshot package is migrated with the Replace processing action, the Migration Manager adds entries into the DOCINFO table, if entries do not exist in the underlying target environment database. The Migration Manager updates existing entries into the DOCINFO table and, finally, attempts to delete those entries in the DOCINFO table that are not present in the XML document that is contained within the migration package.

If the Target production environment contains living attached documents so that the entry in DOCINFO is linked to an actual application record, such as a service request or a purchase order, the DOCINFO business object validation rule prevents any attempt to delete that DOCINFO record. This situation results in the Snapshot migration package failing deployment with the error that was shown previously.

You can use the following preventive measures:

- Perform the initial migration of DOCTYPES before the production goes live. This step ensures that there are no living attached documents that are tied to an application record.
- Perform all Snapshot migrations with the AddModify processing action before and after production goes live. This option ensures that the Migration Manager does not attempt to delete DOCINFO records that are present in the Target database but not present in the XML document that is processed during deployment.

16.1.11 Query migration fails with notunique error

During product implementation, a query is defined against the Work Order Tracking application. The query is marked public and its owner is an administrative user of the system. Query is migrated successfully to the target. This query is deleted and a second query is defined against the same application. The second query also is public and has the same owner but its clause is different. Table 16-1 shows the first and second queries.

Table 16-1 Successive queries defined in source

Query Name	Public?	Application	Owner	Clause
WO_APPR	Yes (1)	WOTRACK	MAXADMIN	(status = 'APPR')
WO_APPR	Yes (1)	WOTRACK	WILSON	(status like '%APPR%')

The deployment of the package that contains the second query fails with the following error:

```
psdi.util.MXApplicationException: signature#notunique
 at psdi.app.signature.FldQueryPublic.validate(FldQueryPublic.java:67)
```

By design, there can be at most one public query for an application with the same query name. In the source environment, no errors were encountered as the first query was deleted before the second query was created. However, in the target environment, the first query still exists when the second is deployed.

There are several recovery options following the failure of the deployment. Because the deployment failed as a result of data in the package, Migration Manager populated the Deployment Data Errors user interface that is giving implementer the opportunity to correct the error. One approach is to find the first query in the target and clear the **Public?** option. In the Migration Manager application, click the Continue Deployment option to complete deployment of the package with the second query. Another approach is to edit the failing XML and set the ISPUBLIC tag to 0(zero), which indicates that the incoming query is not a public query. A third approach is to delete the first query (perhaps it is no longer needed by the implementation) and then continue deployment of the failed package.

16.2 Methods to solve migration package failures

Recovery from migration package deployment failures depends upon the type and complexity of the error that is encountered. We can categorize recovery into the following types:

- Recovering from package failures that result from data validation errors
- Recovering from package failures that result from broader system errors

It might be possible to recovery quickly from data validation errors by using Migration Manager's Error correction and package reprocessing functionality. This approach was described with examples in Chapter 15, "Common topics" on page 351. Implementers should be able to recognize situations where such an approach is valid and effective.

Broader system errors go beyond error correction. For example, if a system configuration is dependent on another configuration and package deployment fails because of the missing configuration, this issue cannot be resolved by correcting the failing package data. A catch up or fix package must be prepared in the source environment and migrated to target before the deployment of the failed package is completed.

Irrespective of the approach that is taken, a detailed product log with statements that cover the deployment benefits the implementers and supports organizations that must resolve deployment failures.

16.2.1 Use of the Logging application

The following sections describe the Logging application.

Migration Manager Guide

The *Migration Manager Guide* describes how to access and set up the Logging application to create log files for you to use in system analysis when an error is encountered. This section describes how to analyze the log files and instructs you to take various actions that are based on what information the log file contains.

For more information about the *Migration Manager Guide*, see this website:

`http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/gp_migmgr/c_migration_logger.html`

Reading the log file

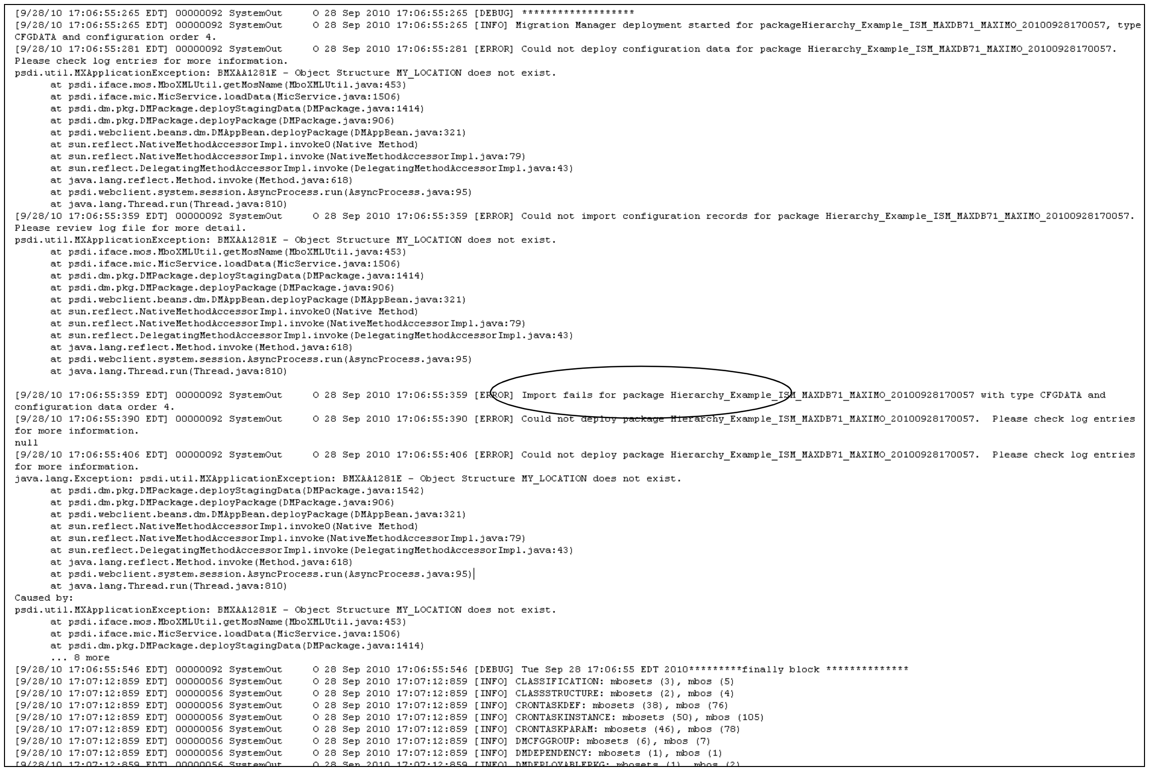
When you encounter an error in the migration, immediately open the log file that you set up in the logging application. When you open the log file, browse to the end and work backwards to find the error. The log file looks similar to Figure 16-5 on page 435. Normally, an error creates several stack traces that you can identify out of the standard log file entries. When you encounter the error, look carefully for text that is similar to the file that is shown in Example 16-1.

Example 16-1 Log file

```
[9/28/10 17:06:55:359 EDT] 00000092 SystemOut     O 28 Sep 2010 
17:06:55:359 [ERROR] Import fails for package 
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057 with type CFGDATA 
and configuration data order 4.
[9/28/10 17:06:55:390 EDT] 00000092 SystemOut     O 28 Sep 2010 
17:06:55:390 [ERROR] Could not deploy package 
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057.  Please check log 
entries for more information.
```

The following aspects of the log file are important:

- The phrase "`Import fails for package`" identifies the section of the log that alerts you to the location of the error in the package.
- The package file identifies that directs you to the offending file is in this statement, as shown in Figure 16-5.

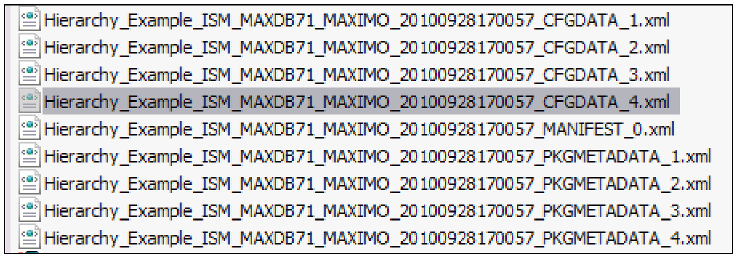


```
[9/28/10 17:06:55:265 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:265 [DEBUG] ******************
[9/28/10 17:06:55:265 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:265 [INFO] Migration Manager deployment started for packageHierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057, type
CFGDATA and configuration order 4.
[9/28/10 17:06:55:281 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:281 [ERROR] Could not deploy configuration data for package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057.
Please check log entries for more information.
psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
      at psdi.iface.mos.MboXMLUtil.getMosName(MboXMLUtil.java:453)
      at psdi.iface.mic.MicService.loadData(MicService.java:1506)
      at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1414)
      at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:906)
      at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:321)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:79)
      at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
      at java.lang.reflect.Method.invoke(Method.java:618)
      at psdi.webclient.system.session.AsyncProcess.run(AsyncProcess.java:95)
      at java.lang.Thread.run(Thread.java:810)
[9/28/10 17:06:55:359 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:359 [ERROR] Could not import configuration records for package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057.
Please review log file for more detail.
psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
      at psdi.iface.mos.MboXMLUtil.getMosName(MboXMLUtil.java:453)
      at psdi.iface.mic.MicService.loadData(MicService.java:1506)
      at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1414)
      at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:906)
      at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:321)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:79)
      at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
      at java.lang.reflect.Method.invoke(Method.java:618)
      at psdi.webclient.system.session.AsyncProcess.run(AsyncProcess.java:95)
      at java.lang.Thread.run(Thread.java:810)

[9/28/10 17:06:55:359 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:359 [ERROR] Import fails for package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057 with type CFGDATA and
configuration data order 4.
[9/28/10 17:06:55:390 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:390 [ERROR] Could not deploy package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057.  Please check log entries
for more information.
null
[9/28/10 17:06:55:406 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:406 [ERROR] Could not deploy package Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057.  Please check log entries
for more information.
java.lang.Exception: psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
      at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1542)
      at psdi.dm.pkg.DMPackage.deployPackage(DMPackage.java:906)
      at psdi.webclient.beans.dm.DMAppBean.deployPackage(DMAppBean.java:321)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:79)
      at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
      at java.lang.reflect.Method.invoke(Method.java:618)
      at psdi.webclient.system.session.AsyncProcess.run(AsyncProcess.java:95)
      at java.lang.Thread.run(Thread.java:810)
Caused by:
psdi.util.MXApplicationException: BMXAA1281E - Object Structure MY_LOCATION does not exist.
      at psdi.iface.mos.MboXMLUtil.getMosName(MboXMLUtil.java:453)
      at psdi.iface.mic.MicService.loadData(MicService.java:1506)
      at psdi.dm.pkg.DMPackage.deployStagingData(DMPackage.java:1414)
      ... 8 more
[9/28/10 17:06:55:546 EDT] 00000092 SystemOut     O 28 Sep 2010 17:06:55:546 [DEBUG] Tue Sep 28 17:06:55 EDT 2010*********finally block **************
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] CLASSIFICATION: mbosets (3), mbos (5)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] CLASSSTRUCTURE: mbosets (2), mbos (4)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] CRONTASKDEF: mbosets (38), mbos (76)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] CRONTASKINSTANCE: mbosets (50), mbos (105)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] CRONTASKPARAM: mbosets (46), mbos (78)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] DMCFGGROUP: mbosets (6), mbos (7)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] DMDEPENDENCY: mbosets (1), mbos (1)
[9/28/10 17:07:12:859 EDT] 00000056 SystemOut     O 28 Sep 2010 17:07:12:859 [INFO] DMDEPLOYABLEPKG: mbosets (1), mbos (2)
```

Figure 16-5 Reading the log file and finding the error

Interpreting the error

Now that you know the location for the file, open the Migration Package with a compressed file editor, and then, open the offending XML file, as shown in Figure 16-6.



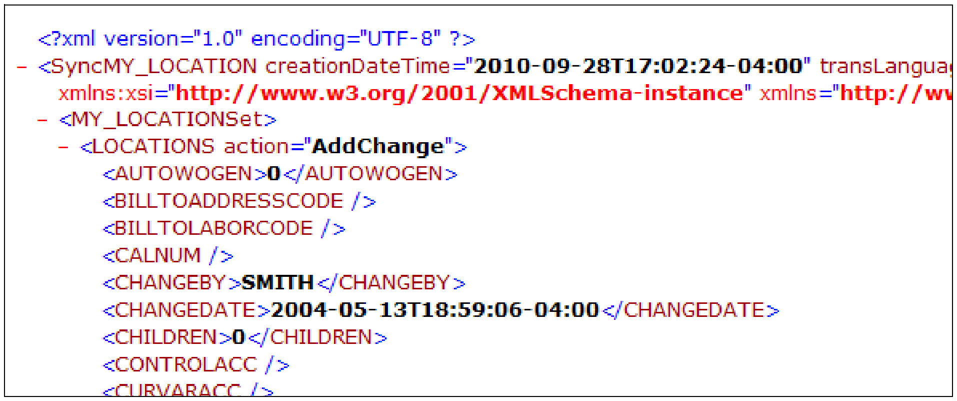
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_CFGDATA_1.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_CFGDATA_2.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_CFGDATA_3.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_CFGDATA_4.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_MANIFEST_0.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_PKGMETADATA_1.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_PKGMETADATA_2.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_PKGMETADATA_3.xml
Hierarchy_Example_ISM_MAXDB71_MAXIMO_20100928170057_PKGMETADATA_4.xml

Figure 16-6 Configuration file with error identified

So, as instructed in the log file, CFGDATA number 4 is where the error lies.

Analyzing the solution

Open the XML file in your editor or viewer of choice. In Figure 16-7, you can see how the XML looks by using Internet Explorer.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SyncMY_LOCATION creationDateTime="2010-09-28T17:02:24-04:00" transLangua
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://ww
  - <MY_LOCATIONSet>
    - <LOCATIONS action="AddChange">
        <AUTOWOGEN>0</AUTOWOGEN>
        <BILLTOADDRESSCODE />
        <BILLTOLABORCODE />
        <CALNUM />
        <CHANGEBY>SMITH</CHANGEBY>
        <CHANGEDATE>2004-05-13T18:59:06-04:00</CHANGEDATE>
        <CHILDREN>0</CHILDREN>
        <CONTROLACC />
        <CURVARACC />
```

Figure 16-7 Examining the XML file

Refer again to the original error, as shown in Example 16-2.

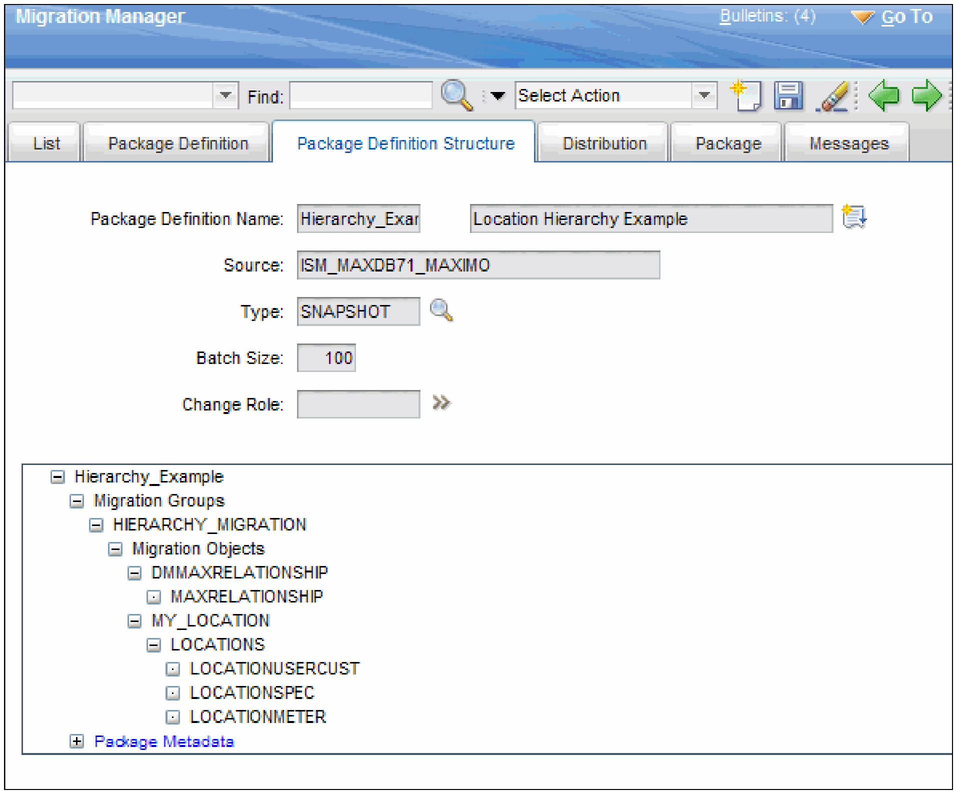
Example 16-2 Original error

```
BMXAA1281E - Object Structure MY_LOCATION does not exist.
```

At the top of the XML file, the action is to the SYNC the Object Structure that is called MY_LOCATION.

When a synchronization action takes place, the system requires that the data is in the system or, at least, has a validation reference point.

In this case, the validation failed because the Migration Manager did not yet process the Object Structure. The data was not part of the package or the processing order was incorrect. In this case, the Object Structure definition was not included, as shown in Figure 16-8.



Package Definition Name: Hierarchy_Exar Location Hierarchy Example

Source: ISM_MAXDB71_MAXIMO

Type: SNAPSHOT

Batch Size: 100

Change Role:

```
⊟ Hierarchy_Example
  ⊟ Migration Groups
    ⊟ HIERARCHY_MIGRATION
      ⊟ Migration Objects
        ⊟ DMMAXRELATIONSHIP
          ⊡ MAXRELATIONSHIP
        ⊟ MY_LOCATION
          ⊟ LOCATIONS
            ⊡ LOCATIONUSERCUST
            ⊡ LOCATIONSPEC
            ⊡ LOCATIONMETER
  ⊞ Package Metadata
```

Figure 16-8 Package definition structure that is missing migration object definition

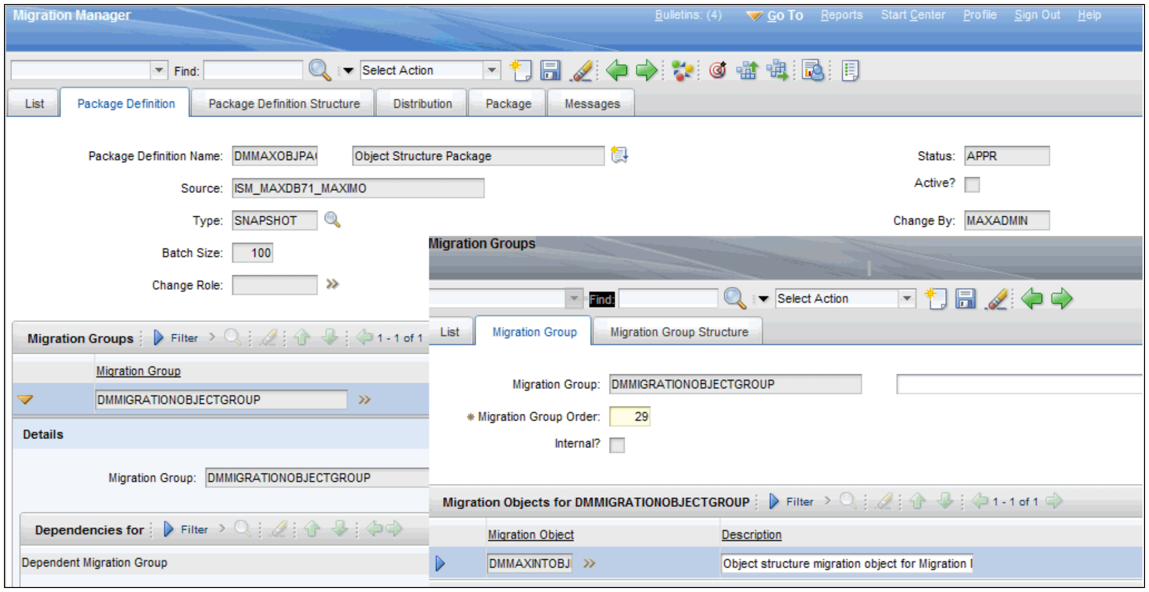
This situation occurs when new object structures are created and not migrated as part of the Data Dictionary migration. You can remedy this situation in one of the following ways:

- You can create a dependency to the migration group for the data dictionary. In this isolated scenario, this approach is time consuming and not recommended.
- You can add the object structures that are required to your migration package definition. By using this technique, you can migrate discrete data artifacts without migrating the entire database. However, the use of this solution causes this same error because how the migration groups are processed does not include a refetch to the processed data, so the package fails again.

Applying the solution

The solution is to create a separate package for only the DMMAXINTOBJ object structure. Create a migration group with this structure alone. Then, create a package with only that new group. After these steps are complete, you can then migrate the package separately before the solution package is migrated.

Figure 16-9 shows how to create this package.



Package Definition Name: DMMAXOBJPA(Object Structure Package
Source: ISM_MAXDB71_MAXIMO
Type: SNAPSHOT
Batch Size: 100
Change Role:
Status: APPR
Active?
Change By: MAXADMIN

Migration Groups 1 - 1 of 1
Migration Group: DMMIGRATIONOBJECTGROUP

Details
Migration Group: DMMIGRATIONOBJECTGROUP

Dependencies for
Dependent Migration Group

Migration Group: DMMIGRATIONOBJECTGROUP
Migration Group Order: 29
Internal?

Migration Objects for DMMIGRATIONOBJECTGROUP 1 - 1 of 1

Migration Object	Description
DMMAXINTOBJ	Object structure migration object for Migration I

Figure 16-9 Object structure migration group

This approach demonstrates the flexibility and power of the Migration Manager and its use when you must troubleshoot various migration scenarios.

16.3 Techniques to prevent migration package failures

In this section, we describe the ways in which you can use the knowledge that is presented so far to prevent migration package errors. The knowledge that is presented in this publication is cumulative to this point, and we hope that you shave successful migration experiences with this information.

16.3.1 Start with identical environments

To whatever extent possible, always have identical environments. We strongly recommend that you refrain from trying to migrate between differing environments from a product standpoint. Migrating between differing environments most often applies to secondary content installations.

16.3.2 Ensure that you understand migration group and object structure relationships

The object structures that are presented to you with the product are thoroughly tested and work as designed. However, as the scenarios that are presented in this book demonstrate, often you migrate portions of data and not all of the packages all at one time. IBM presents you with a complete package with an entire object structure that consists of all dependencies. However, this configuration does not mean that you must process all packages with all dependencies.

When you create your packages, do so with the understanding that dependent data can be migrated first in a separate package. When you create the packages (and the previously migrated configurations are not changed), there is no need to migrate that data again, which is what occurs when you include that migration group as a dependency.

Taking this view helps to prevent failures and increases the speed of your migrations.

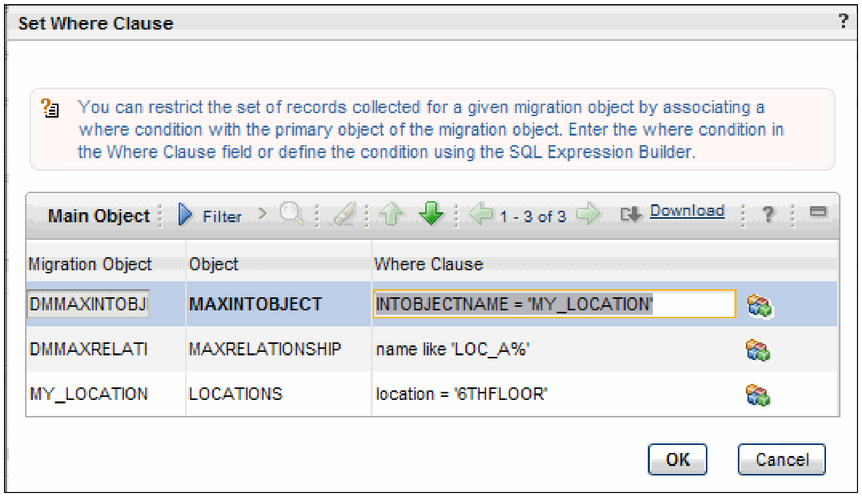
16.3.3 Making separate packages work in a dependent fashion

We described how to understand the structural relationship between migration groups. With this understanding, you can create separate packages for each of the various groups that require sequential processing to quickly and accurately migrate small portions of configuration data.

16.3.4 Using the SQL Where clause for package groups

When you assemble all of the various packages that you require to migrate your configuration changes, the last step is to modify the SQL WHERE clause to limit the package content to only those configuration changes that you want to migrate.

The correct use of the SQL WHERE clause ensures that you migrate only the data that is required. It also ensures that dependent data groups include the data that is required for correct validation, as shown in Figure 16-10.


Set Where Clause

You can restrict the set of records collected for a given migration object by associating a where condition with the primary object of the migration object. Enter the where condition in the Where Clause field or define the condition using the SQL Expression Builder.

Main Object

Migration Object	Object	Where Clause
DMMAXINTOBJ	MAXINTOBJECT	INTOBJECTNAME = 'MY_LOCATION'
DMMAXRELATI	MAXRELATIONSHIP	name like 'LOC_A%'
MY_LOCATION	LOCATIONS	location = '6THFLOOR'

Figure 16-10 SQL Where clause example

16.3.5 Use Migration Collections to separate data assembly from migration

Migration Collections can be used to prepare system configurations in a granular manner that aligns well with the product implementation cycles. The use of collections also precludes the need to plan and prepare change tracking packages and plan and prepare SQL criteria for snapshot packages. Because collections offer a rule-based approach to identifying dependent configurations, rules can be defined, enabled, or disabled to suit the implementation needs. Collections alleviate the need to go through multiple steps to define the package by providing an action to generate the package definition. Package definitions can be discarded and created again as needed. The comparison functionality of Migration Manager is tightly coupled with collections, which gives implementers an accelerated approach to migration.

16.3.6 Use tools such as error correction and value replacement to accelerate deployments

Deployment tools, such as error correction and value replacement, are designed to maintain the accelerated delivery of system configurations into target environment. Upon encountering a package deployment failure, first determine whether error correction or value replacement can resolve the deployment failure rather than planning on re-creating the package in the source environment. In many cases, it is more expedient to reuse the existing package and continue deployment to completion rather than incurring the cost of re-creating package in source environment.

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publication provides more information about the topic in this document. Note that some publications that are referenced in this list might be available in softcopy only.

- *Migration Use Cases with the Migration Manager,* SG24-7906

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

`http://www.ibm.com/redbooks`

Other publications

The following publications also are relevant as further information sources:

- IBM Maximo Asset Management 7.5 Developing Applications

  `http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/topic/com.ibm.mbs.doc/pdfs/pdf_mbs_devapps.pdf`
- Migration Manager User Guide

  `http://pic.dhe.ibm.com/infocenter/tivihelp/v49r1/index.jsp?topic=%2Fcom.ibm.mbs.doc%2Fgp_migmgr%2Fc_ctr_mig_mgr_over.html`

Online resources

The following website also is relevant as further information sources:

- Object structure support for attached documents

  `http://www-01.ibm.com/support/docview.wss?uid=swg24027858`

Help from IBM

IBM Support and downloads

http://www.ibm.com/support

IBM Global Services

http://www.ibm.com/services

A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server, which is available at the following website:

`ftp://www.redbooks.ibm.com/redbooks/SG248132`

Alternatively, you can go to the following IBM Redbooks website:

`http://www.ibm.com/redbooks`

Select **Additional materials** and open the directory that corresponds to the IBM Redbooks form number, SG248132.

Using the web material

The additional Web material that accompanies this book includes the SG248132.zip file, which contains the migration planning spreadsheet, the script that was used in Chapter 7., “Migrating escalations that include automation scripts” on page 163, and the SQL command file that is referenced in Chapter 16, “Troubleshooting” on page 419.

System requirements for downloading the web material

The web material requires the following system configuration:

- Hard disk space: 10 MB minimum
- Operating System: Windows/Linux

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material `.zip` file into this folder.

Migration Use Cases with the Migration Manager Version 7.5

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Migration Use Cases with the Migration Manager Version 7.5

Learn about Migration Manager implementation strategy

Experiment with various real-life scenarios

Review migration troubleshooting tips

By using the Migration Manager, you can migrate configuration content from one production environment to another. The typical use is to migrate configuration content from a development environment to a test environment and then on to production for the Tivoli process automation engine and its applications, such as IBM SmartCloud Control Desk. The goal of migration is to ensure that your production environment fully meets the needs of your users.

This IBM Redbooks publication is an update of the existing book *Migration Use Cases with the Migration Manager*, SG24-7906, and covers the most common migration use cases with the Migration Manager, including the capabilities that were introduced with Tivoli's process automation engine V7.5. These use cases are only a small subset of the possible migration scenarios that can be performed by the Migration Manager, but they were chosen to be representative of the capabilities of the Migration Manager.

This book is a reference for IT Specialists and IT Architects working on migrating configuration content from one production environment to another by using the Migration Manager.

SG24-8132-00 ISBN 0738437867