integrity™ lifecycle manager

Gateway User Guide

12.2.0.0

Contents

1

Installation

Gateway, which supports importing documents into Integrity Lifecycle Manager, is included and installed with the Integrity Lifecycle Manager client. For client installation information, see “Client Installation” in the Integrity Lifecycle Manager Help Center. This section describes additional Gateway requirements and the configuration of Gateway components.

Additional Gateway Requirements

Third-Party Application Requirements

Gateway does not directly use a third-party application for creating XML source files. However, you need an application such as Adobe Acrobat Pro for PDFs for this purpose.

Oracle Database Requirements

To use Gateway with an Oracle database, PTC recommends some performance tuning:

- Ensure that at least 4000 cursors are configured before installing and using Gateway.
- Increase the `Automatically extend datafile when full (AUTOEXTEND)` setting from 5 MB to 10 MB, which is the recommended size for the Oracle tablespace.
- Run `Gather Optimizer Statistics`.

Parser Use and Configuration

Gateway requires a parser to read content from an external source file and transform it into the Item Interchange Format (IIF). Your system administrator generally assumes responsibility for the parser and its configuration. For more information, see System Administrator Tasks on page 10.

To write the parser, consult the Integrity Lifecycle Manager Item Interchange Format specification provided in Item Interchange Format on page 42.

For information on configuring Gateway to use the parser during import, see Wizard Configuration File on page 49. This particular file also instructs Gateway to use a specific Gateway Configuration file. For more information, see Gateway Configuration File on page 64.

Tip

You can use an external parser to transform the document content into Item Interchange Format and import the IIF file directly in the wizard without transformation.

Exporter Use and Configuration

Gateway requires an exporter to transform the Item Interchange Format (IIF) into an external file format. For more information, see Item Interchange Format on page 42. Your system administrator generally assumes responsibility for the exporter and its configuration. For more information, see System Administrator Tasks on page 10.

For information on configuring Gateway to use the exporter during export, see Wizard Configuration File on page 49. This particular file also instructs Gateway to use a specific Gateway Configuration file. For more information, see Gateway Configuration File on page 64.

Configuring JVM

Gateway runs in its own Java Virtual Machine (JVM) instance. The default heap size values follow:

- Minimum heap size = 8 MB
- Maximum heap size = 128 MB

When working with large documents, PTC recommends increasing the maximum heap size.

For more information about configuring the heap size settings, contact PTC Technical Support.

2

Gateway Overview

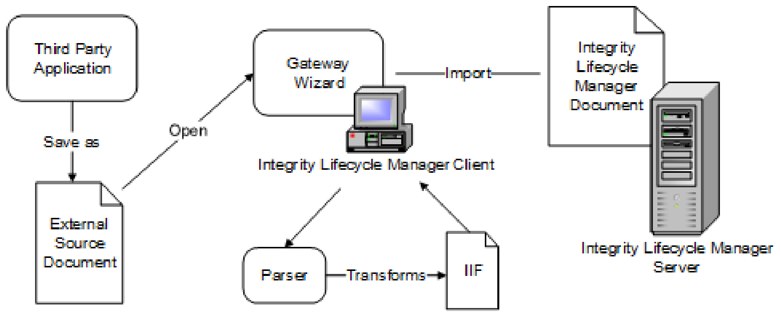
Gateway is a data-driven bridge, allowing imports of external documents into Integrity Lifecycle Manager and exports of Integrity Lifecycle Manager content to external documents.

For imports, it is assumed that the external file with the content to import is an XML-formatted file. However, the source can be in any format if you provide a parser capable of transforming its content into Item Interchange Format (IFF).

IFF describes requirement items, representing requirement items in both the external source file (external items) and in Integrity Lifecycle Manager (internal items). The Gateway Configuration file defines the relationships between the external and internal items.

Gateway Imports

Gateway provides a way to import content from an external source file into Integrity Lifecycle Manager, producing a Requirements document that has the full functionality available in Integrity Lifecycle Manager. Gateway can import from any format if you provide a parser capable of transforming its content to Item Interchange Format (IIF). The available list of parsers is generated based on the file extension of the source file selected in the Gateway Import Wizard. The list of available parsers is specified in the Wizard Configuration file. For more information, see Import Configuration on page 51.


Third Party Application
Save as
External Source Document
Open
Gateway Wizard
Import
Integrity Lifecycle Manager Document
Integrity Lifecycle Manager Client
Parser
Transforms
IIF
Integrity Lifecycle Manager Server

The following is a high-level overview of how to use the Gateway Import Wizard to import content into Integrity Lifecycle Manager:

1. Start the Gateway Import Wizard.
2. Select the parser for transforming the content in the source file to IIF.

   Some Integrity Lifecycle Manager fields require values. For example, a project and category for the document must be specified. If these fields do not exist in the source document, you can map them in the wizard.
3. Specify any additional field mappings that are needed.

Tip

If reimporting the document, specify the Integrity Lifecycle Manager item ID of the document so that the same document is updated and versioned. Otherwise, if an ID is not specified, a new document is created. For more information, see Reimporting Content on page 25.

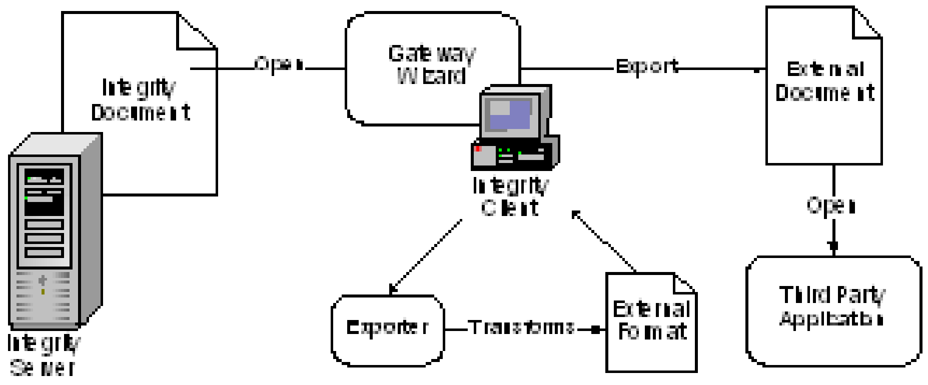
Note

Field mappings are configured in the Gateway Configuration file, and this file is specified in the Wizard Configuration file. Your system administrator generally configures these files for you. For more information, see Gateway Configuration File on page 64. For information about configuring the parser to register and connect with Gateway, see Parser Use and Configuration on page 5.

4. Import the XML file into Integrity Lifecycle Manager.

Gateway Exports

Gateway provides a way to export content from Integrity Lifecycle Manager to an external source document that can then be used in an external editor. Gateway can export to any format if you provide an exporter capable of transforming Item Interchange Format to this format. The available list of exporters is generated based on those specified in the Wizard Configuration file. For more information, see Export Configuration on page 57.



The following is a high-level overview of how to use the Gateway Export Wizard to export content:

1. Open the document with the content that you want to export in Integrity Lifecycle Manager.
2. Start the Gateway Export Wizard. The document is automatically specified in the wizard.
3. Specify the exporter for transforming the document into the desired format.

Note

Field mappings are configured in the Gateway Configuration file, and that file is specified in the Wizard Configuration file. Your system administrator generally configures these files for you. For more information, see Gateway Configuration File on page 64. For information about configuring the exporter to register and connect with Gateway, see Exporter Use and Configuration on page 6.

4. Export the file to an external file, which you can then use in an external editor.

System Administrator Tasks

To simplify imports and exports, your system administrator is generally responsible for setting up the Wizard Configuration file and Gateway Configuration file. The system administrator also generally writes the parser and exporter.

Note

The Gateway Configuration file to use is specified in the Wizard Configuration file.

The following topics provide information about Gateway configurations:

- Parser Use and Configuration on page 5
- Exporter Use and Configuration on page 6
- Wizard Configuration File on page 49
- Gateway Configuration File on page 64
- Remote Loading of Gateway Configuration on page 10

Remote Loading of Gateway Configuration

The system administrator can choose to provide Gateway Configuration files on the Integrity Lifecycle Manager server for Gateway to load remotely. The benefit of a centralized location is that it ensures that all users are using the correct configuration when importing files into Integrity Lifecycle Manager.

If files are stored in the default locations, no configuration is required. However, non-default locations must be specified in the following file:

`installdir/config/properties/is.properties`

where *installdir* is the installation directory for the Integrity Lifecycle Manager server.

For information on configuring `is.properties`, see “Integrity Lifecycle Manager server Configuration Properties” in the Integrity Lifecycle Manager Help Center or the *Integrity Lifecycle Manager Installation and Upgrading Guide*. The following properties affect Gateway:

- `mks.gateway.configdir`

  Specifies the directory for Gateway Configuration files. The default location is:

  `installdir/data/gateway/mappings/`

  where *installdir* is the installation directory for the Integrity Lifecycle Manager server.
- `mks.gateway.tool.configdir`

  Specifies the directory for the Wizard Configuration file. The default location is:

  `installdir/data/gateway/`

  where *installdir* is the installation directory for the Integrity Lifecycle Manager server.

For remote loading of DOCX templates, see Exporter Configuration on page 96 and for remote loading of DSD files, see Importer Configuration on page 103.

Scenario Support

This section includes supported scenarios that illustrate using Gateway with Integrity Lifecycle Manager. The following topics do not exhaustively document all possible actions and are not intended as procedures:

- Import Scenarios on page 11
- Export Scenario on page 14

Import Scenarios

Although all provided import scenarios use requirements from documents as the content to import, other content can be imported. For more information, see Gateway Overview on page 7.

Three supported scenarios follow for using Gateway to import content:

- Import Third-Party Formatted File Scenario on page 12
- Reimport Scenario on page 12
- Import IIF File Scenario on page 13

Import Third-Party Formatted File Scenario

Mary Chang is a team lead with KeyStroke Electronics company, designing a new portable audio player for a customer. The requirements for the player project are provided to the company by the customer in a third-party format.

Mary needs to import the requirements into Integrity Lifecycle Manager so that her team can work from them to create the product.

Before Mary starts, she ensures that she has all files needed to import requirements into Integrity Lifecycle Manager. She obtains a document parser file that matches the type of requirements in the document. The Wizard Configuration file contains a reference to a Gateway Configuration file, which maps all of data from the file to fields in Integrity Lifecycle Manager.

Mary starts the Gateway Import Wizard and selects the file. Mary also specifies the document parser provided to her to read the document data.

Because some data is needed for Integrity Lifecycle Manager that was not in the source file, Mary specifies this information. In this case, the information is the project `MediaPlayer` and the category `System Requirements`.

Now that all the data is inputted into the wizard, Mary runs a validation on the information and reviews it for correctness. There are no errors. Mary continues and views a preview of the information being imported into Integrity Lifecycle Manager. Because this is a first-time import, all of the list items are marked as additions to Integrity Lifecycle Manager.

Mary is satisfied that the information is correctly represented and runs the import operation. The import proceeds, and the results display.

Mary then opens the document in Integrity Lifecycle Manager and uses those requirements to assign and track work for her team.

Reimport Scenario

Patrick Molinas is a lead software architect with SuperSoft company. His product manager notifies him that the requirements from which he is working to design his software application have changed. Patrick needs to update the requirements document stored in Integrity Lifecycle Manager with the changes.

The product manager gives Patrick an XML file that was outputted from Adobe Acrobat Pro. This file contains the requirements that Patrick needs to reimport into Integrity.

Before Patrick starts, he obtains a document parser file that matches the type of requirements in the document. The Wizard Configuration file contains a reference to a Gateway Configuration file, which maps all data from the file to fields in Integrity Lifecycle Manager.

Patrick starts the Gateway Import Wizard and selects that XML file. Patrick also specifies the document parser provided to him to read the document data.

Because some data is needed for Integrity Lifecycle Manager that was not in the source file, Patrick specifies that information. In this case, the information is the project `HappyApp` and the category `Software Requirements`. Also, because this is a document that was previously imported into Integrity Lifecycle Manager, Patrick enters the Integrity item ID for the document. This updates the document on import, rather than creating a new document.

With all the data inputted into the wizard, Patrick runs a validation on the information and reviews it for correctness. There are no errors. Patrick views a preview of the imported information. For a reimport, there are not only additions to Integrity Lifecycle Manager but also changes. Patrick also notices some content deletions, which are(marked with a minus sign (-) symbol). After confirming with his product manager, he decides they are acceptable.

Patrick is satisfied that the information is correctly represented and runs the import operation. The import proceeds, and the results display.

Patrick opens the document in Integrity Lifecycle Manager and notices that the document is versioned. The reimported requirements are the current version, and the original requirements are a previous version. Patrick uses the new version of the document to assign and track work for his team.

Import IIF File Scenario

Wendy Patterson is a software architect at AnyTime Solutions. Wendy’s manager previously used an external parser to create an IIF file containing the requirements for the project and gave the file to Wendy. Wendy needs to import the requirements from this file into Integrity Lifecycle Manager so that she can use them to design the software application.

Wendy starts the Gateway Import Wizard and selects that IIF file. Because the file is in IIF format, Wendy specifies the default document parser, `Sample IIF Importer`.

Because some data is needed for Integrity Lifecycle Manager that was not in the source file, Wendy specifies that information. In this case, the information is the project `HydroApp` and the category `Software Requirements`.

Now that all the data is inputted into the wizard, Wendy runs a validation on the information and reviews it for correctness. There are no errors. Wendy continues and views a preview of the imported information. Because this is an initial import, all of the list items are marked as additions to Integrity Lifecycle Manager.

Wendy is satisfied that the information is correctly represented and runs the import operation. The import proceeds, and the results display.

Wendy then opens the document in Integrity Lifecycle Manager and uses those requirements to create items to track her work and product design.

Export Scenario

Neil Singh is an analyst for KBJ Communications. He needs a copy of an Integrity Lifecycle Manager document to work with a word processor and send to clients.

Neil opens the document in the Integrity Lifecycle Manager client and then starts the Gateway Export Wizard. The document ID for the document he was viewing is already specified in the wizard.

Neil specifies an exporter for the file format he needs (DOCX). The wizard then retrieves the document information from the Integrity Lifecycle Manager server. Neil reviews the document to ensure that it contains the information he intends to export.

There are five warnings in the document. Neil reviews the log file and determines that no action is required to resolve the warnings.

Neil then exports the document to the DOCX format. The wizard indicates that export was successful. Neil can now use the exported file.

Sample Files

Integrity Lifecycle Manager provides server-side and client-side sample files to assist with import and export operations. These sample files are installed with the Integrity Lifecycle Manager server and client. They are also available from the PTC Integrity eSupport portal found at http://www.ptc.com/support/integrity.htm.

Note

- Sample files are intended for use with commonly implemented solution configurations. If you need to modify files for use with your Integrity Lifecycle Manager implementation, it is important to copy them, creating new files with different names. This prevents a future Integrity Lifecycle Manager update from overwriting your modified files if sample files are updated later in service packs or product releases.
- When copying and renaming a sample Gateway Configuration file for use in your system, you must edit the mapping name in the XML header.

For information on sample DOCX templates included with Gateway, see Sample DOCX Templates on page 96.

Integrity Lifecycle Manager server-side Samples

Sample Gateway Configuration files are installed with Gateway in the following location:

`*installdir*/data/gateway/mappings/`
where *installdir* is the installation directory for the Integrity Lifecycle Manager server.

The following sample is included and self-documented: `IIF_Document.xml`.

To use a sample Gateway Configuration file, configure it to match your specific installation.

Integrity Lifecycle Manager client-side Samples

Sample Gateway Configuration files are installed with Gateway in the following location:

`*installdir*/config/gateway/mappings/`
where *installdir* is the installation directory for the Integrity Lifecycle Manager client.

The following samples are included and are self-documented:

- `Document_Export.xml`
- `MS_Word_Requirements_Document_ExternalID.xml`
- `Sample_Requirement_Document_Import_Reimport.xml`
- `Sample_Test_Suite_Import_Reimport.xml`
- `MS_Word_TestSuite_Document.xml`

3

Using the Gateway Wizard

This section provides more detailed information on using the Integrity Lifecycle Manager Gateway wizards for importing and exporting content.

Note

For information on functionality available through the command line interface, see Command Line Interface on page 34.

Importing Content

You can use the Gateway Import Wizard to import content into Integrity Lifecycle Manager. This guide assumes that you are importing content from an external XML file. However, you can also import other types of source files. For more information, see Gateway Overview on page 7.

The following topics provide key information needed to import content from an external document:

- Key Import Considerations on page 17
- Importing Content From an External Document on page 19
- Reimporting Content on page 25
- Troubleshooting Import Errors and Warnings on page 28

Key Import Considerations

Before using the Gateway Import Wizard, note the following:

- Run only one import at a time. If an import is in progress, do not start a second import.
- If the Integrity Lifecycle Manager client is connected to multiple servers, Gateway selects the server from the current view that launches the Gateway Import Wizard. If no view is open, Gateway selects the default server specified in the client preferences. The information about the server that imports the content is displayed in the title bar of the Gateway Import Wizard in the following format:

  Integrity Gateway Import Wizard – `<username>@<hostname>:<port>`

  where:

  - `<username>` is the currently logged in user name.
  - `<hostname>` is the Integrity Lifecycle Manager server hostname.
  - `<port>` is the Integrity Lifecycle Manager server port number.
- When using the Gateway Import Wizard, you must select **Import directly to server, committing all changes** when importing a new document. You also select this option if you want to reimport changes to an existing document directly to the Integrity Lifecycle Manager server. When you click **Import** in the **Preview** pane, the document is saved directly to the server.
- The other import option, **Create pending import on client, allowing for a review of all changes in the Document View**, is valid only when reimporting an existing document. When this option is selected, you can select the checkbox for overwriting any existing pending import already existing for this document on your client machine. When you click **Import** in the **Preview** pane, the pending import is saved to a temporary location on your client machine so that you can review the document before committing it to the server.

- During processing, the Gateway Import Wizard stores temporary copies of IIF files and attachment directories in the system `Temp` directory as configured for your system. By default, the location is:

  `C:\Documents and Settings\`*`UserName`*`\Local Settings\Temp`
  where *UserName* is the currently logged in user name.

  For Linux clients, the location is `/tmp`.

  These temporary files are deleted when you exit the wizard. The directories are not deleted.
- Gateway supports importing subdocuments, which are documents contained in documents as a nested hierarchy.
- To change the `MKS ID` value, manually edit the IIF file. Alternatively, if the input data is suitable, use `REQUIREMENT_ID` values from the document and enable ID priming. To enable ID priming, you change the Gateway Configuration file to use a content link-field.
- PTC recommends configuring Integrity Lifecycle Manager to prevent users from editing fields in items affected by reimporting a file. Contact your Integrity Lifecycle Manager administrator for assistance or consult the *Integrity Lifecycle Manager Installation and Upgrading Guide*.
- Integrity Lifecycle Manager supports populating fields on creation of items. However, rules affect when fields can be updated. When reimporting a file, all configured fields are available for data input. However, Integrity Lifecycle Manager ignores field edits that cannot be applied during the operation. For example, Integrity Lifecycle Manager ignores the **Project** field when reimporting content. Based on your specific Gateway implementation, other field edits can be ignored.
- When viewing a large document in Integrity Lifecycle Manager at the same time you are using Gateway to import a large document, a `JavaHeapSize` error can occur. Large image files are a possible cause. To reduce the risk of error, exit the Integrity Lifecycle Manager client before starting the import or increase the maximum heap size setting for the Integrity Lifecycle Manager client. For more information on setting the maximum heap size, search for "heap size" in the Integrity Lifecycle Manager Help Center.
- Gateway bypasses Integrity Lifecycle Manager field size limits when importing documents, permitting limits to be exceeded. Ensure Integrity Lifecycle Manager field sizes are large enough to accommodate content from imported documents.

Importing Content from an External Document

The following procedure demonstrates how to use the Gateway Import Wizard to import content into Integrity Lifecycle Manager from an external XML-formatted file. For information on running the import command from the command line interface, see Gateway Import Command on page 34.

1. To start the Gateway Import Wizard, select **Document ▸ Import**.

   Note

   If this command is not visible in your viewset, you can add it. For more information, see “To set command preferences in the GUI” in the Integrity Lifecycle Manager Help Center.

   The Gateway Import Wizard starts and displays the **Select Configuration** pane.

2. For **File**, select the file that you want to import.

   Tip

   You can select a ZIP file containing an IIF file named `item.iif` at its root.

   Based on the extension of the selected file, import configurations are listed. The wizard generates this list using import configuration information in the Wizard Configuration file. For more information, see Import Configuration on page 103.

   The description for the configuration that is currently selected is displayed below the list.

3. Select the configuration you want to use to read the file content.

   Tip

   For IIF files, **Sample IIF Importer** is available in the commented section of the server-side Wizard Configuration file.

4. For **Import Options**, select one of the following:
   - **Import directly to server, committing all changes**—Saves imported or reimported content directly to the Integrity Lifecycle Manager server.
   - **Create pending import on client, allowing for a review of all changes in the Document View**—Saves reimported content from a Microsoft Word document to a temporary location on your client machine so that you can review the document before committing it to the server. This option is valid only when you are reimporting a DOCX file. For more information, see Reimporting Content on page 25.
5. In the **Fields** area, specify any necessary field mappings. Fields in red with an asterisk (*) are mandatory.

   This is required when the file selected for import does not include fields for which Integrity Lifecycle Manager requires values. You can specify values for the Integrity Lifecycle Manager fields here. The available mappings are derived from the Wizard Configuration file. For more information, see Wizard Configuration File on page 49.

   Following are the examples of some of the fields:
   - **Category** specifies the document type for the new document.
   - **Document Name** specifies the name to be assigned to the new document
   - **Integrity Document ID** specifies the ID of the root document item. The value is required only when reimporting a document. For more information, see Reimporting Content on page 25. For new documents, do not enter a value. An ID is automatically assigned on a fresh import.
   - **Project** specifies the Integrity Lifecycle Manager project to be assigned to the document on import.

   A data filter panel is displayed for the fields which has `<allowed-values>` element defined in the Wizard Configuration file. For more information on `<allowed-values>` element, see XML Elements for the Wizard Configuration File on page 51.
6. When finished, click **Next**. If you are importing a file other than IIF, the **Creating IIF from input file** window opens. When the IIF file is created, the **Select Content** pane displays.
7. From the **Select Content** list, select the document to import.

   To continue, click **Next**. After the data is validated, the **Validation** pane displays.
8. Under **Review**, confirm the information specified in previous steps.
9. Under **Validation**, verify that the validation step is successful. The validation step verifies that each field marked as required in the Gateway Configuration file contains a value.

If the validation step finds errors, click **Back** and correct the problems. For more information, see Troubleshooting Import Errors and Warnings on page 28.

To continue, click **Next**.

If you are reimporting a document, the document ID is already specified in the previous step. When reimporting, Gateway evaluates the source document and determines if the content matches with the existing Integrity Lifecycle Manager file.

The **Preview** pane displays.

10. Review the document information on both the **Summary** and **Review** tabs.

    - The **Summary** tab provides sum values for various types of changes. A warning displays if the import deletes any items. If any items are deleted, a warning message indicates how many items are deleted. The message also notes that important information such as trace relationships are also deleted. Because deletion of an item is an irreversible operation, PTC recommends that you confirm the affected items on the **Review** tab.

      To generate a report, specify a file location and name for **File** under **Output File Location** and then click **Generate Report**. The fields in the report match those on the **Review** tab. The fields are specified in the Wizard Configuration file. For more information, see Wizard Configuration File on page 49. If a generated report with this file name exists, you are prompted to rename the file. If you generate the report anyway, a sequential numerical digit appended is appended to the file name.
    - The **Review** tab provides details about changes.
      - If no document ID was specified, the import is creating a new document and only the **Importing Document** pane is shown. Each content item displays either an added with children icon or added icon to indicate its status. You can expand and collapse items with children and view additional information about the currently selected item in the **Details** pane.
      - If a document ID was specified, the import is revising an existing document and both the **Importing Document** and **Integrity Document** panes are shown. In these panes, each content item displays an icon to indicate its status.

      The **Status** field in the **Details** pane corresponds to the icons shown in the **Importing Document** and **Integrity Document** panes.

    Descriptions follow of all fields in the **Details** pane.

Field	Description	
Status	Type of change to the Integrity Lifecycle Manager document that results from the import. This field corresponds to the icons shown in the **Importing Document** and **Integrity Document** panes.	
	Added	• The added with children icon indicates that content added to the document and there are changes to its children. • The added icon indicates that content added to the document.
	Changed	• The changed with children icon indicates content changes in the document and its children. • The changed icon indicates content changes in the document.
	Moved	• The moved content with children icon indicates content moved in the document to another location in the tree and there are changes to its children. • The moved icon indicates content moved in the document to another location in the tree.
	Moved and changed	• The moved and changed content with children icon indicates content moved and changed in the document and there are changes to its children. • The moved and changed icon indicates content changes in the document and the content moved to another location in the tree.
	Deleted	• The deleted icon with children icon indicates content that has been removed from the document and there are changes to its children. • The deleted icon indicates content that has been removed from the document.
	Unchanged	• The unchanged with children icon indicates that document content is unchanged and there are changes to its children. • The unchanged icon indicates that content is identical in the source

Field	Description
	document selected for importing and the Integrity Lifecycle Manager document.
Integrity ID	Item ID of the content item in Integrity Lifecycle Manager if available. To view the item or document (if the item is mapped to the Integrity Lifecycle Manager Document Model) in Integrity Lifecycle Manager, click the ID. **Caution** When reimporting document data, clicking each **Integrity ID** link for a document opens a new **Document** view. Because loading large documents can be a client resource-intensive operation, consider limiting your use of links.
Field	When a field is selected, the field name as it is specified in the Gateway Configuration file. Fields are displayed if they are specified in the Wizard Configuration file. For more information, see Wizard Configuration File on page 49. Fields are specified using the `<review fields>` element. They appear in the same order as specified, except for the first field presented. The first field is the first rich text field in the list of comparable fields in the Gateway Configuration file used for field mapping. Rich text is displayed as plain text when displayed in the tree.
Importing Value	Displays the value for the selected field in the imported document.
Integrity Value	Displays the value for the selected field in Integrity Lifecycle Manager. **Note** Integrity Lifecycle Manager bookmarks do not display as fully functioning hyperlinks. The bookmarks display in the **Details** pane with a blue outline. However, clicking them does nothing.

If you disagree with the type of change or its content item match, you can override it. Simply right-click the content item, select **Override Status**, and then make a selection. Available status types appear for selection based on the type of change.

In the **Importing Document** pane:

- **Added** can be converted to **Changed**.
- **Changed** can be converted to **Added**.
- **Unchanged** can be converted to **Added**.

In the **Integrity Document** pane:

- **Deleted** can be converted to **Changed**.
- **Changed** can be converted to **Deleted**.
- **Unchanged** can be converted to **Deleted**.

Converting to **Changed** requires a selection from the **Override Status** window to create a new content item match. The **Override Status** window includes its own **Details** pane to assist you in determining if the item that you intend to select is a match.

For ease of use, right-click and select **Expand All** to see the children of all nodes in the document tree.

11. Visually compare the preview with your original source document and make overrides as required. When you are satisfied with the content and that the document is ready for import, click **Import**.

As the import occurs, the **Results** pane displays processing information, including total numbers for the following:

- Processed items—Unique items edited or published to the Integrity Lifecycle Manager server.
- New items—Items added during the import.
- Deleted items—Items deleted during reimport. Deleted items are not applicable for new imports.
- Changed items—Items changed during reimport. Changed items are not applicable for new imports.
- Skipped items—Items not processed, either based on the Gateway Configuration file settings or due to errors in publishing the item or its parent item.
- Encountered errors and warnings—Errors and warnings occurring during the import. When the import finishes, you can see errors and warnings by clicking **View log file**. For more information, see Gateway Logging on page 33 and Troubleshooting Import Errors and Warnings on page 28.

Note

If there is an error or failure while updating, it is possible that some items are not deleted. Such items are not included in the totals for processed, deleted, and skipped items.

In the lower left corner of the wizard, a progress bar displays the percentage of the import that has been completed. Additionally, you can see the estimated time remaining for completing the import. The estimated time changes, depending on information processed while the import is in progress. The estimated time is based on the average time it has taken to process an item.

When the import finishes, you can click **View log file** to see errors and warnings. For more information, see Gateway Logging on page 33.

- If the import is successful, the **Results** pane displays **Success** and the ID of the root document created in Integrity Lifecycle Manager that corresponds to the imported document. When reimporting, the value is the item ID for the document updated and versioned.
- If the import is unsuccessful, the **Results** pane displays **Failed to import the document**. Close the wizard. After resolving the problems that are indicated in the log file, start the import again. For more information, see Troubleshooting Import Errors and Warnings on page 28.

Reimporting Content

When you are using the sample configurations provided, you can reimport content by specifying the Integrity Lifecycle Manager item ID of the existing Integrity Lifecycle Manager document. In the Gateway Import Wizard, you begin by selecting the file to reimport in the **Select Configuration** pane. You then select an import configuration that supports reimporting.

For example, assume that a DOCX file is selected. The **Integrity Document ID** field is shown in the **Fields** area when one of these import configurations is selected: **External ID Import and Re-Import**, **Requirements Document Import and Re-import with Outline Levels**, or **Test Suite Import and Re-import with Outline Levels**. You specify the item ID of the existing document in the **Integrity Document ID** field.

Additionally, in the **Import Options** area, you select one of two possible options:

- **Import directly to server, committing all changes**—Saves the content that is being reimported directly to the Integrity Lifecycle Manager server.
- **Create pending import on client, allowing for a review of all changes in the Document View**—Saves the content being reimported to a temporary location on your client machine so that you can review the document before committing it to the server.

Note

This option is valid only when you are reimporting a DOCX file.

The remainder of this topic assumes that **Create pending import on client, allowing for a review of all changes in the Document View** is selected. When this is the case, you can select the checkbox for indicating whether to overwrite any existing pending import for the specified document. When this checkbox is selected, any existing pending import is overwritten. If you do not select this checkbox and a pending import exists on your client machine, reimporting fails. The error message indicates that a pending import for this document exists.

When you click**Import** in the **Preview** pane, status information indicates that all changes are pending your review.

When you open the pending import in the **Document** view of the Integrity Lifecycle Manager client, you can review these changes. Additionally, you can make additional edits before saving the document to the server. You can do this review and save immediately after reimporting or do it later.

- For information about both importing and reimporting Microsoft Word files, see Importing DOCX Files on page 97.
- For information on opening a pending import in the Integrity Lifecycle Manager client, see "Opening a Pending Import for Multiple-Row Editing" in the Integrity Lifecycle Manager Help Center.

Only one pending import can be in progress at a time for a given document. If you attempt to reimport a Microsoft Word document when another pending import of this document is in progress, the Gateway Import Wizard displays an error. It also displays errors in other situations, such as those that follow:

- The document is already open in the **Document** view.
- The document has recovered changes that must first be saved or discarded.
- The client machine cannot access the temporary location for pending imports.

Key Reimport Considerations

- When reimporting documents, ensure that you select the correct file to import. Gateway does not require the file selected for reimporting to be in the same location or have the same name as was used in a previous import. Be careful that you do not select a file with the wrong content.
- PTC recommends that you configure Integrity Lifecycle Manager to prevent users from editing fields in items affected by reimporting a file. Contact your Integrity Lifecycle Manager administrator for assistance.

- Because Gateway performs HTML tag replacement on unsupported tags, reimporting a file can take longer due to the increased comparison time. Additionally, tag replacement operations appear as changes in the Integrity Lifecycle Manager item history for content.
- When reimporting a document exported from Integrity Lifecycle Manager without usable IDs, Gateway attempts to match textual changes to existing items. This occurs even if these items were moved within the document tree. Larger requirements with changes are easier for Gateway to match than smaller requirements with changes. Formatting changes are not considered comparable differences in large requirements. If items cannot be identified, then Gateway creates new items and drops old ones as needed. Failure to identify an item generally results because there is no ID or the change was not significant enough.
- Samples for reimporting documents are available in Sample Files on page 14.
- When reimporting a parent document, if you do not have access permissions to the subdocument, processing of the subdocument content is skipped. You can import the other content for which you have access permissions.

  The following warning message is logged in the `GatewayApp.log` file for the items that are skipped due to permission issues:

  ```
  Skipping item "XX" because of Exception MKS124283:
  Permission Denied: Item X is not visible due to
  project/type visibility restrictions.
  ```
- When reimporting a parent document, assume that you have access permissions to the subdocument. Also assume that the imported document has a subdocument node but does not have any content under it. Gateway assumes that the user who exported the parent document did not have access permissions to the subdocument. Consequently, any content in the subdocument is not deleted in the Integrity Lifecycle Manager document.

  The following warning message is logged in the `GatewayApp.log` file for the items that are skipped:

  ```
  "Import action has identified a pattern which might be caused due to
  lack of permission to the sub-document during the export.
  Processing of the sub-segment id: ITEMID is skipped."
  ```

  Gateway reimport does not properly account for a subdocument reference switched from insert to include or include to insert. Assume that the subdocument reference in Integrity Lifecycle Manager is switched from insert to include or from include to insert. After exporting a document that contains subdocuments, the switch could result in data loss on reimport.

Troubleshooting Import Errors and Warnings

The following list provides information on troubleshooting import errors in the `Gateway.log` file:

- `Configuration file error: Mandatory field mismatch for field` *`configuration field`*

  Error occurs because a configuration field is marked as required, but no data for that field exists in the content parsed from the source file.

  If the data is not required from the source file, modify the parser configuration so that the field is not mandatory. If the data is required and exists in the document, modify the document parser so that the value is parsed correctly.
- `Data is missing for optional field <configuration field>`

  Warning occurs when the mapping configuration defines a field that is not required and no data exists in the content parsed from the source file. The warning states that there is no data to save to Integrity Lifecycle Manager for this field.
- `Data will be ignored for field` *`configuration field`*

  Warning occurs when a source has provided data for this field, but the mapping configuration is not configured to accept that field. The warning states that there are data fields present in the input file that cannot be saved to Integrity Lifecycle Manager with the current configuration.
- `Field` *`configuration field`* `configured to link input to MKS items, is not present.`

  Error occurs if a link-field that is configured in the Gateway Configuration file does not exist in the content parsed from the source file. Ensure that the fields configured as link-fields are present in the document, even if they are empty or have no value.

  - If there is a value in the source document, ensure that the Wizard Configuration file is configured to ask for the value.
  - If there is no value in the source document, ensure that the parser is generating an empty value for the field.
- General punctuation was found and mapped to `'?'\n" if $line =~ s/\p{General Punctuation}/?/g`

  Warning occurs if UTF-8 characters in the `General Punctuation` Unicode block are not correctly escaped. Modify the parser to escape these characters.
- `The value for field` *`configuration field`* `should not be empty.`

Error occurs with a `Configuration file error: Mandatory field mismatch for field` *`configuration field`* message. This error occurs when a configuration field is marked as required, but no data has been provided by any source. If DEBUG logging is enabled, Gateway attempts to determine the most likely cause of the error.

- `Unable to retrieve previous import data`

  Error occurs on reimport only. Ensure that the document ID entered is correct and that the item has a source IIF file from the previous import attached to it in Integrity Lifecycle Manager.

  Additionally, ensure that the Gateway Configuration file has the `prior-data-field` setting and `prior-data-name` setting configured correctly.

Exporting Content

You can use the Gateway Export Wizard to export an Integrity Lifecycle Manager document to an external file format. The following topics provide key information needed to export an Integrity Lifecycle Manager document:

- Key Export Considerations on page 29
- Exporting Content from an Integrity Document on page 30

Key Export Considerations

Before using the Gateway Export Wizard, note the following:

- Run only one export at a time. If an export is in progress, do not start a second export.
- If the Integrity Lifecycle Manager client is connected to multiple servers, Gateway selects the server from the current view that launches the Gateway Export Wizard. If no view is open, Gateway selects the default server specified in the client preferences. The information about the server that exports the content is displayed in the title bar of the Gateway Export Wizard in the following format:

  Integrity Gateway Export Wizard - `<username>@<hostname>:<port>`

  where:

  - `<username>` is the currently logged in user name.
  - `<hostname>` is the Integrity Lifecycle Manager server hostname.
  - `<port>` is the Integrity Lifecycle Manager server port number.
- Gateway supports exporting subdocuments that are children in the hierarchy of the root document that you are exporting. For information on subdocuments, see the *Integrity Lifecycle Manager Help Center*.

- If you do not want other users editing the exported document, ensure that the mapping file permits edits on import only by your user ID. When you set this at the time of the first export, only your changes are included on import of the document.

Caution

If you permit edits by other users, it is possible for your data in the document to be lost. Corresponding Integrity Lifecycle Manager items are then deleted on import.

To configure the mapping file, refer to the `owner` property in the `<set property>` element.

- For `Double`, `Float`, `Integer`, and `Long` field types, Gateway uses the actual field value, not the display value. This can affect an import of a document that was exported.
- While exporting a parent document, if you do not have access permissions to a subdocument, you can export the parent document. However, the subdocument content ID, name, description, and state are not exported because you do not have access permissions to the subdocument.

Exporting Content from an Integrity Document

The following procedure demonstrates how to use the Gateway Export Wizard to export content from an Integrity Lifecycle Manager document to an external file. For information on running the export command from the command line interface, see Gateway Export Command on page 38.

1. Select the Integrity Lifecycle Manager document that you want to export.
2. To start the Gateway Export Wizard, select **Document ▸ Export**.

Note

If this command is not visible in your viewset, you can add it. For more information, see “To set command preferences in the GUI” in the *Integrity Lifecycle Manager Help Center*.

The Gateway Export Wizard starts and displays the **Configure** pane.

3. Under **Configuration**, specify the following:
   - **Integrity Item IDs** specifies the Integrity Lifecycle Manager item number of the root item for the document that you are exporting. The document itself determines the value for this field. This value is only editable if the wizard is started from the command line interface. For more information, see Gateway Export Command on page 38).
   - **Export Configuration** specifies the export transformer file to be used to create the output document file. The configuration in the Wizard Configuration file determines this list. For more information, see Export Configuration on page 57.
4. Under **Options**, specify the following:
   - **Filter** specifies the filter query definition to be applied on the exported document. The available options are:
     - **None**—Does not filter the exported document. This is the default setting.
     - **User Specified**—Applies a filter that you define. You can define the filter in the Integrity Lifecycle Manager Document view. For more information on the Document view, see the *Integrity Lifecycle Manager Help Center*. Filters can also be defined using the Gateway command line interface. For more information, see Gateway Export Command on page 38.
     - **From Configuration**—Automatically applies the filter that is defined through the Gateway configuration file. For more information on the required XML syntax for the filter query definition, see `<filter-query-definition>` in Export Configuration on page 57.

     Note

     - To view the filter that Gateway is to apply, click **Show Definition**. The **Filter Definition** window opens, displaying the definition. If you need to save that definition, you can manually copy the text to another file. To close the window, click **OK**.
     - When applying a filter query definition to a document, you cannot use a text-only filter. However, you can use a filter that specifies **"any text field contains"**.

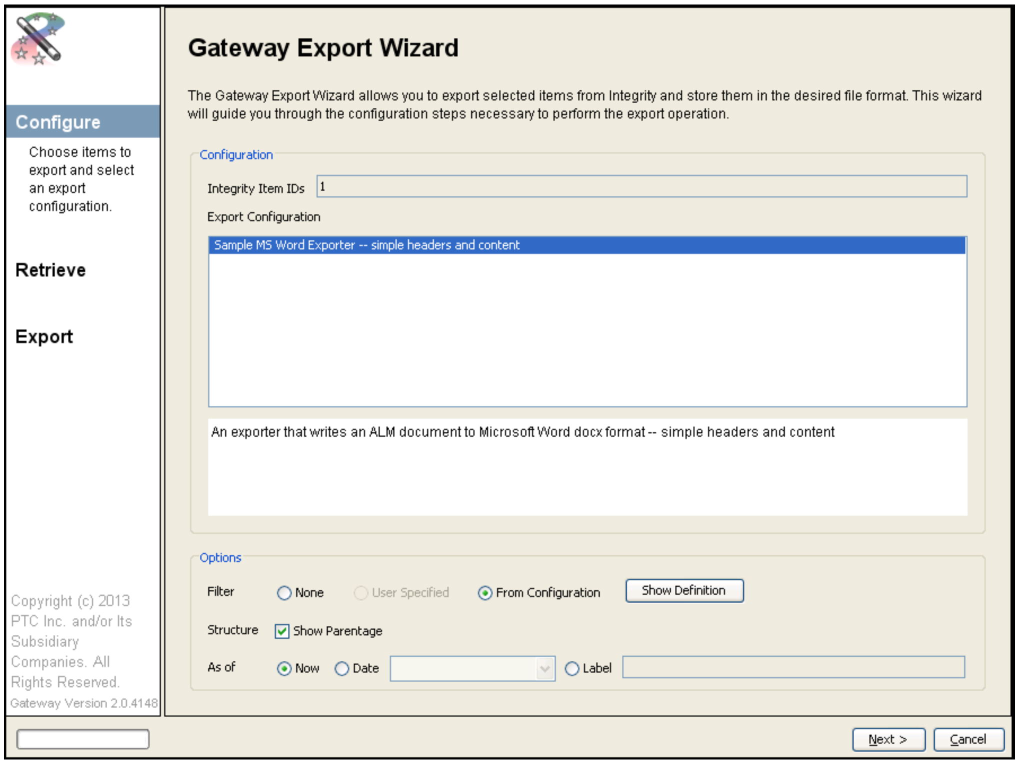
   - **Structure** specifies the **Show Parentage** option. When enabled, all ancestors in the exported document are shown, from leaf nodes to the document root. To use this option, you must also specify a filter. By default, parentage is not shown.

If the **Document** view is configured to show parents in filtered documents and a filtered is applied, the **Show Parentage** option is available.

The **Show Parentage** option can also be pre-set through the Gateway Configuration file. For more information on the required XML syntax for showing parentage, see Export Configuration on page 57.

- **As of** specifies a previous version of the document to export, allowing you to export a historical document as of a specific date or label. If no value is specified, the current date and time is used.

The following panel shows a sample export configuration:



Gateway Export Wizard

The Gateway Export Wizard allows you to export selected items from Integrity and store them in the desired file format. This wizard will guide you through the configuration steps necessary to perform the export operation.

Configure

Choose items to export and select an export configuration.

Retrieve

Export

Configuration

Integrity Item IDs: 1

Export Configuration: Sample MS Word Exporter -- simple headers and content

An exporter that writes an ALM document to Microsoft Word docx format -- simple headers and content

Options

Filter: None User Specified [✓ From Configuration]

Structure: [✓ Show Parentage]

As of: [✓ Now] Date Label

Copyright (c) 2013 PTC Inc. and/or Its Subsidiary Companies. All Rights Reserved.
Gateway Version 2.0.4148

5. To continue, click **Next**. The **Retrieve** pane displays the progress as the document is retrieved from Integrity Lifecycle Manager. Once the document is retrieved, the **Retrieve** pane displays a summary of the items contained in the document.
6. Review the summary of the items contained in the document. You can also review a log of any errors or warnings that occurred during retrieval. To resolve any problems, see Troubleshooting Import Errors and Warnings on page 28.
7. To continue, click **Next**. The **Export** pane displays.
8. Review the export configuration and summary information. You want to ensure that you are satisfied with it before starting the actual export.

9. For**Select File**, specify the file name to which you are exporting the document. The default file name is of the form:

   ```
   itemID - date.fileExtension
   ```
   where

   - *itemID* is the ID of the document (root Integrity Lifecycle Manager item) that you are exporting.
   - *date* is the date of export.
   - *fileExtension* is the file type.

   For example, assume that the file name is `64073 - 2015-04-09.iif`.
10. When finished, click **Export**. The document is exported to the specified file.

Note

- If you specified a filter, the resulting exported document includes only the content segments defined in that filter.
- If you specified an external script exporter as your export configuration, additional windows could open.

11. To view the export log, click **View log file**. For more information, see Gateway Logging on page 33.

There are currently no known issues with re-exporting documents using Gateway. Documents are re-exported in the same way that they are initially exported. If you export to the same file, the exporter configuration determines if the file is overwritten or content is appended. For more information, see Exporter Use and Configuration on page 6.

Gateway Logging

Gateway provides a log file that contains information useful for troubleshooting problems. You can view the log from the **Results** page of the Gateway wizard by clicking **View log file**. The log file is displayed in the default editor configured for your operating system. For example, assume that you default editor is Windows Notepad.

The log file is generated in the following default location:

```
installdir\bin\Gateway.log
```
where *installdir* is the location the Integrity Lifecycle Manager client is installed.

By default, Gateway is configured to maximum logging levels. If desired, you can turn on debug. To configure `Gateway.log`, contact PTC Technical Support.

Command Line Interface

This section provides information on the commands available from the command line interface.

Note

Currently, Gateway does not return the necessary information to the console to make it suitable for scripting purposes. However, some information is written to the Gateway log. For more information, see Gateway Logging on page 33. To make full use of this information, Gateway must be started from the Integrity Lifecycle Manager client GUI.

Gateway Import Command

The Gateway import command imports content into Integrity Lifecycle Manager. For detailed information on importing content from an external file, see Gateway Imports on page 8 and Importing Content From an External Document on page 19.

Caution

Run only one import at a time. If an import is in progress, do not start a second import.

From the command line, type:

```
Gateway import [--user=value] [--password=value] [--hostname=value]
[--port=value] [--config=value] [--file=value] [--fields=value]
[--fieldsDelim=value] [--pendingimport] [--overwritepending] [--silent] [--importMode]
```

where:

`--user=value`
Specifies the user name to connect to the Integrity Lifecycle Manager server.

`--password=value`
Specifies the user password to connect to the Integrity Lifecycle Manager server.

`--hostname=value`
Specifies the name (alias or IP address) of the Integrity Lifecycle Manager server hosting the documents.

`--port=value`
Specifies the port number of the Integrity Lifecycle Manager server hosting the documents.

`--config=value`
Specifies the name of the parser to use. For more information, see Parser Use and Configuration on page 5.

`--file=value`
Specifies the file containing content to import into Integrity Lifecycle Manager. The file must be one of the following:

- Integrity Lifecycle Manager Item Interchange Format (IIF) file
- Zipped (ZIP) file containing an Integrity Lifecycle Manager IIF named `item.iff` at the root.
- Any external file containing content that must be parsed on import. For more information, see Parser Use and Configuration on page 5.

`--fields=value`
Specifies field mapping pairs. By default, field separators are semi-colon (;) delimited. An example follows:

```
--fields="fieldname1=fieldvalue1[;fieldname2=fieldvalue2[;...]]"
```

If semicolons appear in the input, use the `--fieldsDelim` option. Field mappings are specified in the Gateway Configuration file. For more information, see Gateway Configuration File on page 64.

Note

If reimporting a document, specify the ID of the Integrity Lifecycle Manager item that corresponds to the imported document. For example, specify `--field=MKS_DOCUMENT=34338`.

`--fieldsDelim=value`
Specifies a substitute field delimiter instead of the default semi-colon. Use with `--fields` option. An example follows:

```
--fieldsDelim="|"
--fields="project=project1|category=category1"
```

`--pendingimport`
Specifies the creation of a pending import in a local directory on the client machine rather than committing the document directly to the server. This option is valid only for reimporting a Microsoft Word document.

`--overwritepending`
Specifies that if a pending import exists when reimporting a Word document, it is to be overwritten with a newly created pending import.

`--importMode` specifies that items created for the import are done by `im importissue`, rather than `im createissue`, using the Integrity Lifecycle Manager API.

Locating Traces Dropped During a Gateway Reimport

The IDs of items that are added, deleted, changed, or skipped during reimport are logged in the Gateway log. For more information, see Gateway Logging on page 33. You can use this information to develop a script to locate the relationships and traces that are dropped during the reimport.

1. Note the time before starting the reimport. Use the following command to get the time:
   ```
   im diag --diag=dbtime
   ```
2. Import the document. For more information, see Gateway Import Command on page 34.
3. Parse the `Gateway.log` file.
4. Locate the traces that are dropped for deleted items by executing this command and parsing the response:
   ```
   mksapiviewer --iplocal im relationships --asOf="date prior
   to reimport" --expandLevel=0 itemId1, itemId2, …..
   ```

Example

The following example provides information about the Integrity Lifecycle Manager commands that can be used to locate the dropped traces. These commands can be used in scripts. Their output needs to be parsed to obtain the relevant information.

1. Get the time before reimport:
   ```
   im diag --diag=dbtime
   ```
   The output is:
   ```
   Jun 3, 2015 5:37:11 PM
   ```
2. Import the document.

   The following is the output of a sample `Gateway.log` file.
   ```
   Input file successfully loaded and parsed.
   Input file has 4 items.
   Successfully retrieved prior data.
   Requirements Document will be re-imported to '962'.
   Imported 'Requirements Document': ID 962
       Number of processed items: 5
       Number of new items: 0
       Number of deleted items: 1
       IDs of deleted items: 978
       Number of changed items: 4
       IDs of changed items: 972,962,974,976
       Number of skipped items: 0
   ```
3. Check the `Number of deleted items` string. If the value is non-zero, check the `IDs of deleted items` string. Read the IDs.

In this sample `Gateway.log` file, the value for the `IDs of deleted items` string is 978.

4. Locate the traces that are dropped for the deleted items by using the following command:

```
mksapiviewer --iplocal im relationships --asOf="3 Jun, 2015
5:37:11 PM" --expandLevel=0 978
```

Note

The output of the `im diag --diag=dbtime` command returns the date and time in the format `MMM d, yyyy h:mm:ss a`. The `im relationships` command requires `asOf` to be in the format `d MMM, yyyy h:mm:ss a`. The date format depends on the client locale.

The output is:

```
Response:
  App. Name     = im
  Command Name = relationships
  Work Item:
    Id          = 978
    Context     = NULL
    Model Type = im.Issue.content
    Field:
      Name       = Shares
      Data Type = mksItemList
      ItemList:
    Field:
      Name       = Modelled By
      Data Type = mksItemList
      ItemList:
    Field:
      Name       = Summary
      Data Type = NULL
      Value      = NULL
    Field:
      Name       = Satisfied By
      Data Type = mksItemList
      ItemList:
    Field:
      Name       = Validated By
      Data Type = mksItemList
      ItemList:
    Field:
      Name       = Contains
      Data Type = mksItemList
      ItemList:
    Field:
        Name      = Spawns
        Data Type = mksItemList
        ItemList:
      Field:
        Name      = References
        Data Type = mksItemList
        ItemList:
          Item:
            Id         = 979
            Context    = 978
            Model Type = im.Issue.Relationship.Out
      Field:
        Name      = Decomposes To
        Data Type = mksItemList
        ItemList:
      Field:
        Name      = Verified By
        Data Type = mksItemList
        ItemList:
          Item:
            Id         = 99
            Context    = 978
            Model Type = im.Issue.Relationship.Out
      Field:
        Name      = Is Related To
        Data Type = mksItemList
        ItemList:
      Field:
        Name      = ibplfield
        Data Type = mksItemList
        ItemList:
    Exit Code    = 0
```

By parsing the response, you can conclude that the following relationships are dropped:

- `Name: References;ID: 979`
- `Name: Verified By;ID: 99`

Gateway Export Command

The Gateway export command exports an Integrity Lifecycle Manager document to a specified file. For more information, see Gateway Exports on page 9 and Exporting Content on page 29.

⚠ **Caution**

Run only one export at a time. If an export is in progress, do not start a second export.

From the command line, type:

```
Gateway export [--user=value] [--password=value] [--hostname=value]
[--port=value] [--config=value] [--file=value]
[--filterQueryDefinition=value][--[no]showParentage][--asOf=value]
[--silent] documentID...
```

where:

`--user=value`
Specifies the user name to connect to the Integrity Lifecycle Manager server.

`--password=value`
Specifies the user password to connect to the Integrity Lifecycle Manager server.

`--hostname=value`
Specifies the name (alias or IP address) of the Integrity Lifecycle Manager server hosting the documents.

`--port=value`
Specifies the port number of the Integrity Lifecycle Manager server hosting the documents.

`--config=value`
Specifies the name of the export configuration to use when exporting a document. This configuration is specified in the Wizard Configuration file, which is named `gateway-tool-configuration.xml`.

`--file=value`
Specifies the file name to which to save the document.

`--filterQueryDefinition=value`
Specifies the filter query definition to apply on exporting a document. Value syntax is dependent on the specific command shell. For example, in MKS Kornshell, you specify:

```
--filterQueryDefinition="(field[Category] = Imported Requirement)"
```

If the defined query excludes the document root, then Gateway creates an empty root. If the parent of the node is not exported, then the node is attached to the document root. This means that the grandchild is not attached to the grandparent if the parent is not exported. You can modify the behavior of this option using the `--showParentage` option.

Note

If you encounter quoting issues when specifying a query definition in your command shell, you can use the `--filterQueryDefinitionFile` option.

`--filterQueryDefinitionFile`
Specifies the name of the file where the filter query definition is stored. This option is useful if you are encountering quoting issues in the command shell that you are using. You can also use this option for larger filter query definitions.

`--[no]showParentage`
Specifies whether ancestor information is shown when a document is exported. If the `--showParentage` option is applied, then all ancestors, from leaf nodes to the document root, are included in the exported document. To use the `--showParentage` option, you must also set a filter value. By default, parentage is not shown.

`--asOf=value`
Specifies a previous version of the document to export (exporting a historical document) as of a specific date or label. If no value is specified, the current date and time is used. Possible values include:

- `label:value`
  Specifies a label. For example, the following specifies the document version as of label `ABCrelease1`:

  ```
  --asOf=label:ABCrelease1
  ```

- `:now`
  Specifies a document as of the current date and time.
- A date string in your default locale format. For example, when using the US locale, you specify the date string in one of the following formats:

  ```
  "MM/dd/yyyy h:mm:ss a z"
  "MM/dd/yyyy h:mm:ss.SSS a z"
  "MM/dd/yyyy h:mm:ss a"
  "MM/dd/yyyy h:mm:ss.SSS a"
  "MM/dd/yyyy"
  ```

`--silent`
Specifies not to start the Gateway Wizard GUI for input and status messages. If this option is not specified, the GUI displays by default.

`documentID...`
Specifies the item ID of the Integrity Lifecycle Manager document to export. To specify multiple IDs, separate each ID with a space.

4

Item Interchange Format

The Item Interchange Format (IIF) is an XML-based format for describing requirements items. IIF is used to represent requirements items in the external source data (external items) as well as in standard Integrity Lifecycle Manager requirements items (internal items). The Gateway Configuration file defines the relationship between the external and internal items.

XML Elements in IIF Documents

`<MKS Items>`

The `<MKS Items>` element is the highest level XML element in an IIF document, containing all other elements. There is only one `<MKSItems>` element per IIF document.

- `Attributes`: Because the `<MKSItems>` element is the root XML element, it must define a number of attributes that relate to the IIF schema. These attributes identify the XML namespaces used with the IIF document and specify the schema version.
  - `xmlns`
    Specifies the namespaces used in the IIF file and identifies the set of XML tags used with prefixes. This documentation and the sample IIF document use the following `xmlns` attributes:
    ```
    xmlns="http://www.mks.com/schemas/MKSItems"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    ```
    These attributes mean that the MKS IIF tags described are the ones used without the prefix. Any XHTML tags used in the document must have an `xhtml:` prefix. However, if you are creating an IIF document that contains many XHTML tags, you could have the MKS IIF tags require a `mks:` prefix and allow XHTML tags without prefixes. To do so, use the following `xmlns` attributes instead:
    ```
    xmlns:mks="http://www.mks.com/schemas/MKSItems"
    xmlns="http://www.w3.org/1999/xhtml"
    ```
  - `schemaVersion`
    Specifies the version of the schema used by the IIF document. The current schema version is 1.0.
- `Parent Element`: None (highest level XML element)
- `Sub-Elements`: `<Metadata>, <Item>`

`<Metadata>`

The `<Metadata>` element contains information about the IIF document. This element is optional.

- `Attributes`: None
- `Parent Element`: `MKSItems`
- `Sub-Elements`: `<Date>, <IntegrationId>`

`<IntegrationId>`

The `<IntegrationId>` element specifies a unique integration identifier. This element is optional.

- `Attributes`: None
- `Parent Element`: `<Metadata>`
- `Sub-Elements`: None

`<Item>`

The `<Item>` element describes an item. An item consists of one or more fields and can contain subitems or children that are described using the `<Item>` element.

- `Attributes`
  - `prototype`
    Specifies the type of item being defined. Values for prototype are:
    - `DOCUMENT`—A document is being defined.
    - `CONTENT`—Content of a document is being defined.
  - `id`
    Specifies an identifier unique to an external system. This attribute is optional.
- `Parent Element`: `<MKSItems>, <Children>`
- `Sub-Elements`: `<Field>, <Richcontent>, <Relationship>, <Attachments>, <Children>`

`<Field>`

The `<Field>` element describes a basic data field.

- `Attributes`
  - `name`
    Specifies the name of the field.
  - `dataType`
    Specifies the type of data that the field contains. Values include:
    - `string`
    - `date`
    - `integer`

    When the `dataType` attribute is unspecified, a data type of `string` is assumed.
- `Parent Element`: `<Item>`
- `Sub-Elements`: None

`<Richcontent>`

The `<Richcontent>` element contains a field that includes text and graphics that are formatted using XHTML. This is the XML-compatible form of HTML. To specify an XHTML tag as part of the content of the field, precede the XHTML tag name with the `xhtml:` tag. Also, all data in this field must be wrapped in a single `<xhtml:div>` ... `<xhtml:div>` tag.

This XHTML content can include links. Links to local resources must be relative to the location of the IIF. It is not required that these resources are explicitly added as attachments in the IIF.

For example, if you include a two-part attachment, such as a Visio diagram with a picture, specify the following link:

```
<xhtml:a href="relativepath/diagram.vsd"><xhtml:img src=
"relativepath/diagram.png"/></xhtml:a>
```

Note

Use the forward slash (/) in paths for attachments.

- `Attributes`
  - `name`
    Specifies the name of the field.
- `Parent Element`: `<Item>`
- `Sub-Elements`: Any XHTML element prefixed with `xhtml:`. For example, `<xhtml:p>` is a sub-element.

  Note

  If you swapped which namespaces require prefixes in the attributes to the `<MKSItems>` element, the `xhtml:` prefix is not required.

`<Relationship>`

The `<Relationship>` element describes a relationship field that defines a relationship with another item. It contains an ordered list of relationship values specified with the `<Target>` element.

- `Attributes`
  - `name`

Specifies the name of the field.

- `Parent Element`: `<Item>`
- `Sub-Elements`: `<Target>`
  The following is an example for specifying the `<Relationship>` element:
  `<mks:Relationship name="fieldname1">` Relationship field name
  `<mks:Target id="11111" />` ID of item to relate
  `<mks:Target id="22222" />` ID of second item to relate
  `</mks:Relationship>`

`<Target>`

The `<Target>` element points to another item by specifying an identifier unique to that field.

- `Attributes`
  - `id`
    Specifies an identifier unique to an external system. The value for this attribute is prefixed with the prototype of the item if the item is `ISSUE` or `DOCUMENT`. No prefix defaults to `CONTENT`.
- `Parent Element`: `<Relationship>`
- `Sub-Elements`: None

`<Attachments>`

The `<Attachments>` element describes a field which contains one or more file attachments.

- `Attributes`
  - `name`
    Specifies the name of the field.
- `Parent Element`: `<Item>`
- `Sub-Elements`: `<Attachments>`
  The following is an example for specifying the `<Attachments>` element:
  `<mks:Attachments name="Attachments">` field name
  `<mks:Attachment name="image.png">`
  `images/image.png</mks:Attachment>` file name and a path on disk relative to the IIF file
  `</mks:Attachments>`

`<Attachment>`

The `<Attachment>` element describes an individual file attachment.

- `Attributes`

    - `name`
      Specifies the file name of the attachment. This file name is relative to the IIF document.
- `Parent Element`: `<Attachments>`
- `Sub-Elements`: None

`<Children>`

The `<Children>` element holds additional `<Item>` elements that are subordinate to or part of the current `<Item>`. The next topic provides the XML schema for IIF documents. In this schema, the `<Item>` with the prototype of `DOCUMENT` contains several children, each with the prototype of `CONTENT`.

- `Attributes`: None
- `Parent Element`: `<Item>`
- `Sub-Elements`: `<Item>`

XML Schema for IIF Documents

Following is the XML schema that defines the IIF XML tags.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:MKSItems="http://www.mks.com/schema/MKSItems"
           targetNamespace="http://www.mks.com/schema/MKSItems"
           xmlns:xhtml="http://www.w3.org/1999/xhtml"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
    <xs:import namespace="http://www.w3.org/1999/xhtml"
schemaLocation="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd" />
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
 schemaLocation="http://www.w3.org/2001/xml.xsd"/>
 <xs:element name="MKSItems">
  <xs:annotation>
    <xs:documentation>This is the root element.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Metadata" type="MKSItems:Metadata"
minOccurs="0"maxOccurs="1" />
      <xs:element name="Item" type="MKSItems:Item"
minOccurs="1"maxOccurs="unbounded" />
  </xs:sequence>
    <xs:attribute name="schemaVersion" type="xs:string" use="optional" />
</xs:complexType>
</xs:element>

<xs:complexType name="Item">
```

```
  <xs:annotation>
    <xs:documentation>Item representationxs:documentation>Item
  representation> </xs:annotation>
  <xs:sequence>
     <xs:element name="Field" type="MKSItems:Field" minOccurs="0"
maxOccurs="unbounded"/>
     <xs:element name="Richcontent" type="MKSItems:Richcontent"
minOccurs="0" maxOccurs="unbounded"/>
     <xs:element name="Relationship" type="MKSItems:Relationship"
minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Attachments" type="MKSItems:Attachments"
minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Children" type="MKSItems:Children" minOccurs="0"
maxOccurs=" 1"/>
  </xs:sequence>
  <xs:attribute name="prototype" type="xs:string" use="optional" />
  <xs:attribute name="mksid" type="xs:unsignedShort" use="optional" />
  <xs:attribute name="id" type="xs:string" use="optional" />
</xs:complexType>

  <xs:complexType name="Metadata">
   <xs:annotation>
     <xs:documentation>Optional metadata for the ItemXML.xs:documentation>
  </xs:annotation>
  <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1"
 name="Date" type="xs:dateTime" />
      <xs:element minOccurs="0" maxOccurs="1"
name="IntegrationId" type="xs:string">
  </xs:sequence>
</xs:complexType>
```

5

Wizard Configuration File

This chapter provides documentation on configuring the Gateway Wizard to use parsers and exporters.

- Parsers transforming data from an external source into IIF.
- Exporters transforming IIF data from Integrity Lifecycle Manager to an external format.

The Wizard Configuration file is an XML file named `gateway-tool-configuration.xml`. This file is editable in the following locations:

- `installdir/config/gateway/gateway-tool-configuration.xml`, where *installdir* is the installation directory of the Integrity Lifecycle Manager client.
- `installdir/data/gateway/gateway-tool-configuration.xml`, where *installdir* is the installation directory of the Integrity Lifecycle Manager server.

The two files are merged in memory to provide a logical union for use by Gateway.

During future upgrades, your changes to the Wizard Configuration file are preserved.

Note

- The Wizard Configuration file is currently at version 1.5.
- `<query>` element can be used with the `<field>` attribute to filter the results of a query. For more information, see XML Elements on page 51.

XML Elements for the Wizard Configuration File

`<configuration>`

The `<configuration>` element contains the individual `<parser-configs>` and `<export-configs>` elements. There is only one `<configuration>` element in the Wizard Configuration file.

- `Attributes`
  - `version`
    Specifies the version of the schema used. For example, the version is 1.3.
- `Parent Element`: None (highest level XML element)
- `Sub-Elements`: `<parser-configs>, <export-configs>`

`<parser-configs>`

The `<parser-configs>` element contains the individual `<parser-config>` elements, each of which can specify a configuration for a given document parser.

- `Attributes`: None
- `Parent Element`: `<configurations>`
- `Sub-Elements`: `<parser-config>`

`<export-configs>`

The `<export-configs>` element contains the individual `<export-config>` elements, each of which can specify a configuration for a given document exporter.

- `Attributes`: None
- `Parent Element`: `<configurations>`
- `Sub-Elements`: `<export-config>`

Import Configuration

`<parser-config>`

Each `<parser-config>` element contains the information that describes a document parser.

There is one `<parser-config>` element for each definable wizard configuration.

- Parent Element: <configurations>
- Sub-Elements: <name>, <description>, <parser>, <extension>, <display-fields>,<config-fields>, <gateway-configuration-name>

<name>

The <name> element specifies the name of the document parser. The Gateway Wizard displays this name in the list of available document parsers when it prompts for the selection of a document parser. Each <parser-config> element contains one <name> element.

- Attributes: None
- Parent Element: <parser-config>
- Sub-Elements: None

<description>

The <description> element specifies a brief description of the document parser. The Gateway Wizard displays this description in the list of available document parsers when it prompts for the selection of a document parser. Each <parser-config> element contains one <description> element.

- Attributes: None
- Parent Element: <parser-config>
- Sub-Elements: None

<parser>

The <parser> element specifies how the document parser is implemented. Each <parser-config> element contains one element.

- Attributes
  - id
    Specifies the type of Java object to create to run the document parser. Values for id include:
    - SCRIPT—The document parser is implemented as an external script. This id value requires the following command property and value pair:
      ```
      <property name="command">value</property>
      ```
      The command property specifies the external script run by the Java object created to implement the document parser. This script is written in any language. The content of the element is the actual command line run by the Java object. To indicate the file to transform and the output file, include {0} and {1}, respectively, where these items appear in the command line.

    - `IIF`—The document parser assumes that the external source is already expressed in IIF and reads the external data unaltered.
  - `Parent Element`: `<parser-config>`
  - `Sub-Element`: `<property>`

`<property>`

The `<property>` element specifies configuration information to pass to the exporter script.

- `Attribute`: `name` specifies the name value pair of the property. An example follows:
  `<property name="command">value</property>`
- `Parent Element`: `<parser>`
- `Sub-Elements`: None

`<extensions>`

The `<extensions>` element contains a list of default file extensions for the type of file handled by the document parser. The Gateway Wizard uses this list to determine the list of available document parsers presented. Each `<parser-config` element contains one `<extensions>` element.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: `<extension>`

`<extension>`

The `<extension>` element specifies a default file extension for the document parser. There is one `<extension>` element for each default file extension.

- `Attributes`: None
- `Parent Element`: `<extensions>`
- `Sub-Elements`: None

`<display-fields>`

The `<display-fields>` element specifies the field that represents the item title that the Gateway Wizard displays. Each `<parser-config>` element contains one `<display-fields>` element.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: `<display-field>`

`<display-field>`

The `<display-field>` element specifies the item field that represents the title that the Gateway Wizard displays. The `<display-fields>` element contains at least one `<display-field>` element.

- `Attributes`: None
- `Parent Element`: `<display-fields>`
- `Sub-Elements`: None

`<config-fields>`

The `<config-fields>` element specifies a list of fields to include in the IIF items that the document parser creates. The input file cannot determine the field values. Each `<parser-config>` element contains one `<config-fields>` element.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: `<config-field>`

`<config-field>`

The `<config-field>` element specifies information about an IIF field that the user specifies. The Gateway Wizard uses this information to present an appropriate method of specifying the IIF field value to include in IIF items created by the document parser. There is one `<config-field>` element for each manually specified IIF field.

- `Parent Element`: `<config-fields>`
- `Sub-Elements`: `<external-name>, <display-name>, <default-value>, <mandatory>,<allowed-values>`

`<external-name>`

The `<external-name>` element specifies the name of the IIF field to manually configure. Each `<config-field>` element contains one `<external-name>` element.

- `Attributes`: None
- `Parent Element`: `<config-fields>`
- `Sub-Elements`: None

<display-name>

The <display-name> element specifies a string to display as the field name in the Gateway Wizard or other presentation layers. This allows you to customize the display name of the field rather than using the <external-name> element, which specifies the internal name of the IIF field.

Each <config-field> element contains at most one <display-name> element. When no <display-name> element is specified, the internal field name specified by <external-name> is shown.

- Attributes: None
- Parent Element: <config-field>
- Sub-Elements: None

<default-value>

The <default-value> element specifies the default value for the IIF field. The Gateway Wizard presents this default value to the user when prompting for the field's value. Each <config-field> element contains one <default-value> element.

- Attributes: None
- Parent Element: <config-field>
- Sub-Elements: None

<mandatory>

The <mandatory> element specifies if the field is required in the IIF items created by the document parser. It can contain either true or false. When the value is true, the Gateway Wizard requires the user to provide a value for this field. Each <config-field> element contains one <mandatory> element.

- Attributes: None
- Parent Element: <config-field>
- Sub-Elements: None

<allowed-values>

The <allowed-values> element specifies a list of allowed values for the IIF field. Each <config-field> element contains one <allowed-values> element.

- Attributes: None
- Parent Element: <config-field>
- Sub-Elements: <value>, <query> - Only one of the sub-elements can be defined in the <allowed-values> element.

`<value>`

The `<value>` element specifies a valid value for the IIF field. The Gateway Wizard presents this value in a list when prompting for the field's value. There is one `<value>` element for each possible value that can be specified for the IIF field.

- `Attributes`: None
- `Parent Element`: `<allowed-values>`
- `Sub-Elements`: None

`<query>`

The `<query>` element specifies a valid value for the IIF field. If more than one query is specified, a consolidated result is displayed in the Gateway Wizard. There is one `<value>` element for each possible value that can be specified for the IIF field.

- `Attributes`: field specifies a valid field name to filter the results of a query. An example follows:

  `<query field="project">`All Projects`</query>`
- `Parent Element`: `<allowed-values>`
- `Sub-Elements`: None

`<gateway-configuration-name>`

The `<gateway-configuration-name>` element specifies the name of the Gateway configuration. This is the name specified by the `<mapping>` element in the Gateway Configuration file, not the name of the file itself. There is one `<gateway-configuration-name>` element for each Gateway configuration used by the document parser.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: None

`<review-fields>`

The `<review-fields>` element specifies the columns (fields) that display in the import preview.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: `<review-field>`

`<review-field>`

The `<review-field>` element specifies the field name that displays as a column in the import preview.

- `Attributes`: None
- `Parent Element`: `<review-fields>`
- `Sub-Elements`: None

Export Configuration

`<export-config>`

Each `<export-config>` element contains the information that describes a document exporter.

There is one `<export-config>` element for each definable wizard configuration.

- `Parent Element`: `<export-configs>`
- `Sub-Elements`: `<name>, <description>, <exporter>, <extension>, <filter-query-definition>, <display-fields>, <gateway-configuration-name>, <show-parentage>`

`<name>`

The `<name>` element specifies the name of the document exporter file. The Gateway Wizard displays this name in the list of available document exporters. Each `<export-config>` element contains one `<name>` element.

- `Attributes`: None
- `Parent Element`: `<export-config>`
- `Sub-Elements`: None

`<description>`

The `<description>` element specifies a brief description of the document exporter. The Gateway Wizard displays this description in the list of available document exporters when prompted for the selection of a document parser. Each `<parser-config>` element contains one `<description>` element.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: None

<exporter>

The `<exporter>` element specifies how the document exporter is implemented. Each `<export-config>` element contains one `<exporter>` element.

- `Attributes`
  - `id`
    Specifies the type of Java object to create to run the document exporter. Values for `id` include:
    - `SCRIPT`—The document exporter is implemented as an external script. This `id` value requires the following `command` property and value pair:

      ```
      <property name="command">value</property>
      ```

      The `command` property specifies the external script run by the Java object created to implement the document exporter. This script is written in any language. The content of this element is the command line run by the Java object. To indicate the file to transform and the output file, include {0} and {1}, respectively, where these items appear in the command line.
    - `IIF`—The document exporter assumes that the external source is expressed in IIF and reads the external data unaltered.
- `Parent Element`: `<exporter-config>`
- `Sub-Elements`: `<property>`

<property>

The `<property>` element specifies configuration information to pass to the exporter script.

- `Attributes`: `name` specifies the name value pair of the property. An example follows:

  ```
  <property name="template">Path to a template</property>
  ```

- `Parent Element`: `<exporter>`
- `Sub-Elements`: None

<extension>

The `<extension>` element specifies the file extension of the exported file.

- `Attributes`: None
- `Parent Element`: `<export-config>`
- `Sub-Elements`: None

`<filter-query-definition>`

The `<filter-query-definition>` element specifies the filter definition to be applied on an export configuration. The behavior of the `<filter-query-definition>` element can be modified by using the `<show-parentage>` element.

The filter specified here is always available as a filter selection when using the Gateway Export Wizard. In the wizard, you select the **From Configuration** option under the available **Filter** options. A few examples follow.

To export a document according to its priority:

```
<filter-query-definition>(field["Priority"]&lt;&gt;0)</filter-querydefinition>
```

To export a document that shows only the category `Imported Requirements`:

```
<filter-query-definition>(field["Category"] = "Imported Requirement")
</filter-query-definition>
```

To export a document that shows only high or critical items created in the last seven days:

```
<filter-query-definition>((field["Created Date"] in the last 7 days) and
(field["Priority"] = "High","Critical"))</filter-query-definition>
```

- `Attributes`: None
- `Parent Element`: `<export-config>`
- `Sub-Elements`: None

Note

Special characters that are included as part of the filter definition must be escaped. Special characters can include any character that the XML/DTD definition does not allow, such as `<, >, !`. Other string-based filters, such as `(field[Category]=Heading)`, do not contain characters that require escaping.

`<show-parentage>`

The `<show-parentage>` element specifies whether Integrity Lifecycle Manager displays all ancestors for an exported document. When set to `<show-parentage>true</show_parentage>`, all ancestors in the exported document are shown, from leaf nodes to the document root. When set to `<show-parentage>false</show-parentage>`, the document's ancestors are not shown. By default, parentage is not shown.

- `Attributes`: None
- `Parent Element`: `<export-config>`
- `Sub-Elements`: None

Note

To use the `<show-parentage>` element, you must also specify a filter.

`<display-fields>`

The `<display-fields>` element specifies the field that represents the item title that the Gateway Wizard displays. Each `<parser-config>` element contains one `<display-fields>` element.

- `Attributes`: None
- `Parent Element`: `<parser-config>`
- `Sub-Elements`: `<display-field>`

`<display-field>`

The `<display-field>` element specifies the item field that represents the title that the Gateway Wizard displays. The `<display-fields>` element contains at least one `<display-field>` element.

- `Attributes`: None
- `Parent Element`: `<display-fields>`
- `Sub-Elements`: None

`<gateway-configuration-name>`

The `<gateway-configuration-name>` element specifies the name of a Gateway configuration. This is the name specified by the `<mapping>` element in a Gateway Configuration file, not the name of the file. There is one `<gateway-configuration-name>` element for each Gateway configuration used by the document exporter.

- `Attributes`: None
- `Parent Element`: `<export-config>`
- `Sub-Elements`: None

Sample Wizard Configuration File

The following sample Wizard Configuration file shows how the various XML elements described in the previous section work together.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configurations version =”1.5”>
    <parser-configs>
         <name>MS Word Importer</name>
             <description>An importer that imports a MS Word document into Integrity
             </description>
             <parser id="MSWORD-DSD">
                   <property name="xslt">http://integrity_server:7001/gateway/import/
                    Microsoft Word/WordImport.DSD.xslt</property>
             </parser>
             <extensions>
                  <extension>docx</extension>
             </extensions>
             <display-fields>
                  <display-field>Title</display-field>
             </display-fields>
             <config-fields>
                  <config-field>
                        <external-name>Title</external-name>
                        <display-name>Document Name</display-name>
                        <mandatory>true</mandatory>
                        <default-value>Requirements Document</default-value>
                        <allowed-values/>
                  </config-field>
                  <config-field>
                        <external-name>Project</external-name>
                        <display-name>Project</display-name>
                        <mandatory>true</mandatory>
                        <default-value>/ALM_Projects/Release1</default-value>
                        <allowed-values>
                              <query field="Project">ALM_All Projects</query>
                        </allowed values>
                  </config-field>
                  <config-field>
                        <external-name>Category</external-name>
                        <display-name>Category</display-name>
                        <mandatory>true</mandatory>
                        <default-value>System Requirement</default-value>
                        <allowed-values>
                              <value>Comment</value>
                              <value>Heading</value>
                              <value>Business Requirement</value>
                              <value>Component Requirement</value>
                              <value>Functional Requirement</value>
                              <value>Non-Functional Requirement</value>
                              <value>System Requirement</value>
                              <value>Technical Requirement</value>
                              <value>User Requirement</value>
                        </allowed-values>
                  </config-field>
                  <config-field>
                        <external-name>IssueId</external-name>
                        <display-name>Integrity Document ID</display-name>
                        <mandatory>false</mandatory>
                        <default-value/>
                        <allowed-values/>
                  </config-field>
             </config-fields>
             <review-fields>
                  <review-field>Summary</review-field>
                  <review-field>Category</review-field>
             </review-fields>
             <gateway-configuration-name>Field Mapping Configuration ID
             </gateway-configuration-name>
       </parser-config>
    </parser-configs>
   <export-configs>
       <export-config>
           <name>Sample Document Exporter</name>
           <description>a sample document exporter</description>
           <exporter id="SCRIPT">
              <property name="template">template.docx</property>
           </exporter>
           <extension>iif</extension>
           <display-fields>
              <display-field>DOCUMENT_TITLE</display-field>
           </display-fields>
           <show-parentage>false</show-parentage>
           <gateway-configuration-name>sample-parser-config</gateway-
configuration-name>
       </export-config>
    </export-configs>
</configurations>
```

Sample Wizard Configuration for Filtered Export

This topic provides a sample Gateway Wizard Configuration file, which is an XML file named `gateway-toolconfiguration.xml`. This sample file shows how you can configure the required XML elements for a document title and filter query definition.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configurations version="1.5">


              <!-- Import configurations are defined here -->
    <export-configs>
       <export-config>
          <name>Microsoft Word Exporter -- Filtered</name>
          <description>An exporter that writes a filtered ALM document to
          Microsoft Word docx format</description>
          <exporter id="MSWORD">

             <property name="template">../config/gateway/export/
             Microsoft Word/2009-WordExportTemplate-05-Simple.docx
             </property>
          </exporter>
          <extension>docx</extension>
          <display-fields>
             <display-field>Document Title</display-field>
          </display-fields>
          <gateway-configuration-name>MKS Sample ALM Document Export
          </gateway-configuration-name>
          <filter-query-definition>(field["Priority"]&lt;&gt;0)
          </filter-query-definition>
          <show-parentage>true</show-parentage>
      </export-config>
   </export-configs>
</configurations>
```

6

Gateway Configuration File

The Gateway Import Wizard uses a parser and the Item Interchange Format (IIF) to convert data from an external source into Integrity Lifecycle Manager items. Behind the scenes, the Gateway Configuration file accomplishes the following:

- Defines the relationship between items from an external data source, as represented in an IIF document, and internal Integrity Lifecycle Manager items. These relationships allow external item fields to fill in fields in the corresponding Integrity Lifecycle Manager item.
- Provides conditional mappings by using `<map-conditional>` and `<map>` elements

The Gateway Wizard uses the Gateway Configuration file as a method of importing data into Integrity Lifecycle Manager. Other integrations can use the Gateway Configuration file to define two-way relationships between internal and external items. These two-way mappings allow the two sets of items to remain synchronized such that changes made in one set of items are reflected in the other. However, the export of Integrity Lifecycle Manager items to external data sources is beyond the scope of this guide.

The installation of a solution template can include sample Gateway Configuration files. If you are installing a solution, consult the solution documentation for information on the use of the files.

Gateway Configuration files are also included when using the Admin Migration Wizard to migrate an Integrity Lifecycle Manager server configuration from one server environment to another. For more information, see the Integrity Lifecycle Manager Help Center or the*Integrity Lifecycle Manager Installation and Upgrading Guide*.

How the Gateway Configuration File Works

While a configuration is an XML file, similar to an IIF document or the Gateway Wizard Configuration file, it is more than a way of representing data. It is a small programming language that lets you define the process of importing and exporting items from an external data source and Integrity Lifecycle Manager internal items.

The Gateway Configuration file processes items one at a time and determines the complete set of mapping rules to apply to the item before applying them. This allows later steps to change or override previous mapping rules. The original values of the internal and external fields are always used, regardless of the mappings that are scheduled to be applied.

There are two other important concepts for configuration files: conditional mappings and levels.

Conditional Mappings

Conditional mappings are mapping instructions that only apply when a `<map-conditional>` element or `<map-translation>` sub-element activate them. These conditional mappings are contained within `<map>` elements. Until mappings are activated, they are ignored. Once a `<map>` is activated, the mapping instructions within that element become active. This can include additional `<map-conditional>` elements and the corresponding `<map>` elements. However, such `<map>` elements are also ignored until they are activated.

Levels

Levels are a related concept. The process begins with the main instructions at level 1. Each time a `<map>` element is activated, the process moves up a level and remains at that new level until the `</map>` tag is reached. At that point, the process moves back to the previous level.

The importance of levels is in the way they interact with the `<map-conditional>` and `<map>` elements. When a `<map-conditional>` element attempts to find a matching `<map>`, only `<map>` elements at the current level of processing are examined. Assume that you have a `<map-conditional>` element within a `<map>` element. The corresponding `<map>` elements that `<map-conditional>` can activate must also appear within the same `<map>` element.

Additionally, once a `<map-conditional>` on a given level has activated a `<map>` element and its contents, all other `<map-conditional>` elements at the current level are ignored.

XML Elements for the Gateway Configuration File

`<mapping>`

The `<mapping>` element is the highest level XML element, containing all other elements. There is only one `<mapping>` element in the file.

- `Attributes`
  - `name`
    The `name` attribute specifies the name of the particular Gateway conversion process defined in the file. This name is the identifier through which a Gateway configuration is selected for use by the Gateway Wizard. It is expected to be unique across all known Gateway configurations.
  - `template-version`
    The `template-version` attribute specifies the version of the XML template used for creating the Gateway Configuration file. This documentation describes version 3.0. This value is required for Gateway to function correctly.
- `Parent Element`: None (highest level XML element)
- `Sub-Elements`: `<description>`, `<set-property>`, `<link-field>`, `<field>`, `<map-conditional>`, `<map>`

`<description>`

The `<description>` element specifies a description of the conversion process defined in the file. The Gateway Wizard displays this description when prompting for the selection of the Gateway configuration to use.

There is one `<description>` element in a configuration element.

- `Parent Element`: `<mapping>`
- `Sub-Elements`: None

`<set-property>`

The `<set-property>` element sets an item control property to a specified value. An example follows:

```
<set-property name="owner" value="external"/>
```

Some properties available to you when importing items into Integrity Lifecycle Manager are:

- `adapter`

This property specifies to use one of the named Gateway adapters in mapping data. It defines the command set that is used for publishing and retrieving data so that how Gateway knows how to work with Integrity Lifecycle Manager. Valid values include:

  - `IMAdapter`
    Default adapter used for loosely related items, typically retrieved using an Integrity Lifecycle Manager query. This is the fallback adapter used by Gateway.
  - `RQAdapter`
    Adapter used for the Integrity Lifecycle Manager document model. This adapter is used by default in some integrations that explicitly request the use of the document model.
  - `RelationshipTreeAdapter`
    Imports relationship tree-connected items as a contained data set. There is no case where this adapter is the default. The template must explicitly request use of this adapter. For more information, see Using the Relationship Tree Adapter on page 107.

- `allow-restructure`

  When reimporting updated data, this property controls if items can be moved or reordered within the document or relationship tree. Possible values include true and false. If false, the mapped item data is updated according to the template, but the item itself is not moved.

  This property only applies to structural data models. For example, it applies to the Integrity Lifecycle Manager document model and Integrity Lifecycle Manager relationship trees if the adapter property specifies `RelationshipTreeAdapter`.

- `insertMode`

  This property configures Gateway to import the subdocument node as an included document. The property is only applicable to the Integrity Lifecycle Manager document model. It tells Gateway how to import subdocument nodes. Possible values are `reference` and `include`. For information on these two document model concepts, see the *Integrity Lifecycle Manager Help Center*.

- `owner`

  This property helps Integrity Lifecycle Manager determine whether the item is externally or internally owned. Gateway uses this property in its decision to deal with items that are missing from the structured data in the document or relationship tree being imported. If the item is externally owned, Gateway removes Integrity Lifecycle Manager items that are not in the input data from the existing structure in the document or relationship tree. If the item is internally owned, Gateway does not remove these items. Valid values include:

  - `external`
    Specifies that the items primarily occur outside of Integrity Lifecycle Manager and copies are imported into or updated in Integrity Lifecycle Manager. In the case of a document, for example, externally owned data means that Integrity Lifecycle Manager deletes content that is no longer present within the reimported data.
  - `internal`
    Specifies that the items are internally managed and that changes to the items primarily occur in Integrity Lifecycle Manager and copies are exported.

- `prior-data-field`

  When a document is reimported, this property identifies to Integrity Lifecycle Manager the location of the attachment field within an IIF document that represents the previously imported data. This property is only relevant to document items.

- `prior-data-name`

  When a document is reimported, this property identifies to Integrity Lifecycle Manager the name of the attachment file (IIF document) that represents the previously imported data. Previously imported data is used to help identify the set of changes to perform on the data.

- `structural-relationship`

  This property indicates the name of the relationship field in Integrity Lifecycle Manager that represents the top-down structure (parent to child relationship) in a relationship tree. This property is only used when the adapter property specifies `RelationshipTreeAdapter`.

- `custom-field-values`

  This property indicates the name of field in Integrity Lifecycle Manager having `customfieldvalues` data type. The value must be specified when an individual custom field value mapping is used. This property is used to identify that the custom field value mapping is individual.

- `Attributes`

  - `name`
    Specifies the name of the property to set.
  - `value`
    Specifies a value for the named property.

- `Parent Element`: `<mapping>`,`<map>`
- `Sub-Elements`: None

`<link-field>`

The `<link-field>` element automatically creates a link between a unique ID field in the external data source and the internal data. By creating this link, it is easier to update existing fields when reimporting previously imported data.

The `<link-field>` element has four forms. The first form is:
`<link-field field-type="id">`

When using this form, each item in the external data source has a unique identifier, similar to the primary key of a database. This unique identifier is then used as the Integrity Lifecycle Manager identifier.

The second form is:
`<link-field external="REQID" field-type="id">`

In this form, a field in the external data named `REQID` contains the Integrity Lifecycle Manager identifier for the corresponding field in the internal data.

The third form is:
`<link-field internal="DATA_KEY">`

As with the first form, the external data source has a unique identifier. However, instead of using it as the Integrity Lifecycle Manager identifier, it is stored in the internal field named `DATA_KEY`.

The fourth form is:
`<link-field external="REQ_ID" internal="DATA_KEY">`

With this form, the external data field named `REQ_ID` is linked with the internal data field named `DATA_KEY`.

- `Attributes`
  - `external`
    Specifies the name of a field in the external data source
  - `internal`
    specifies the name of a field in the internal data.
  - `field-type`
    Specifies the type of the field. For example, `field-type="id"` specifies that the field is an ID field.
  - `hash-code`
    Specifies the `hash-code` attribute, which specifies a field name in which a hash code based on the identifier in question is stored.
    Some external data sources, such as Oracle databases, do not allow database lookups using text strings. Only integers can be used. To accommodate such sources, use the `hash-code` attribute. To possibly

improve Integrity Lifecycle Manager performance, your database administrator can create an index for the queried field. However, before using this attribute or making any database changes, your DBA must evaluate the impact of the database change to determine if it is suitable for your Integrity Lifecycle Manager implementation.

- `Parent Element`: `<mapping>, <map>`
- `Sub-Elements`: None

`<field>`

The `<field>` element defines the mappings between fields in the external data source and fields in the internal data. The simplest version of this tag names only the external and internal fields. An example follows:

```
<field external="Type" internal="Share Category"/>
```

- `Attributes`
  - `external`
    Specifies the name of a field in the external data source.
  - `internal`
    Specifies the name of a field in the internal data, including user-defined fields and existing fields such as **Section**.
  - `direction`
    Specifies the direction of the relationship between the external data and internal data fields. Values are:
    - `in`—The internal data field is updated from the external data field.
    - `out`—The external data field is updated from the internal data field.
    - `both`—The external and internal data field can be updated from the other, depending on context.
    - `none`—Ignores all previously defined mappings.

    If the direction attribute is unspecified, `direction=both` is the default behavior.
  - `field-type`
    Specifies what type of content the field contains. Values are:
    - `type`—Integrity Lifecycle Manager item type.
    - `relationship`—Integrity Lifecycle Manager relationship field
    - `attachment`—An attachment
    - `richcontent`—Formatted text that can include links and graphics
    - `date`—Date with no time
    - `date-time`—Date and time
    - `number`—Integer or floating point number

    - `ibpl`—Item backed pick list (IBPL)

      Note

      Not all integrations recognize `ibpl` as a valid value for `field-type`. The value of the IBPL field is displayed for the integrations that recognize this value. Currently, Microsoft Word, Project, Excel, and IIF integrations recognize the value of the IBPL field. Edit in Word recognizes only the IBPL field value. For all other integrations, the IDs of the backing items for the IBPL field are displayed.

    - `custom-field-value`—Supports individual custom field value
    - `id`—Integrity Lifecycle Manager document ID

      Note

      When `field-type="id"` is specified in the configuration file with `"direction=out"`, the exported file for the versioned document displays the versioned ID of the Integrity Lifecycle Manager document.

- `data-type`
  Specifies how to interpret the data in the field. Possible values include:
    - `xhtml`—The rich content in a field is in XHTML, not Integrity Lifecycle Manager Rich Content format.
- `attachment`
  Specifies the Integrity Lifecycle Manager attachment field used to store images embedded in rich content fields.
- `external-date-format`
  Specifies the date format used in the external data source. The format `EEE MM/dd/yy` is an example. This attribute is only meaningful if `field-type="date"` or `field-type="date-time"` is also specified; otherwise, it is ignored.
- `external-locale`

Specifies the locale used in the external data source.IETF BCP 47 language tags, such as `de- DE`, are supported. Additionally, ISO639 alpha 2 or alpha 3 language codes optionally concatenated with ISO3166 alpha 2 country codes are supported, such as `en` or `en_US`. If a value is not specified, the default value is the current system locale.

Note

This attribute is meaningful only if `field-type="date"`, `field-type="date-time"`, or `field-type="number"` is also specified. Otherwise, it is ignored.

- `clobber`
  Specifies how existing attachments associated with the field are handled. The `clobber` attribute is only meaningful when `field-type= "attachment"` is also specified; otherwise, it is ignored. Possible values are:
  - `true`—All attachments for the field are replaced with the associated version of the attachment, even if the attachment is unchanged.
  - `false`—Existing attachments for the field are not removed and only updated attachments are modified.

  If the `clobber` attribute is unspecified, `false` is the default behavior.
- `on-create-only`
  Specifies whether the field is only populated on initial creation. Values are:
  - `true`—The field is only populated on its initial creation. If the field exists, it is not populated.
  - `false`—If the field exists, it is populated.
- `value-translation-type`
  Species how to translate data when the external and internal fields store data differently. For example, assume that the external field contains a range of integer values, and the internal field contains string values of `Low`, `Medium`, and `High`. The `value-translation-type` attribute specifies the relationship. Values are:
  - `string-string`—A string value in the external field corresponds to a different string value in the internal field.
  - `intrange-string`—A range of integers in the external field corresponds to a string value in the field.
  - `integer-integer`—An integer value in the external field corresponds to a different integer value in the internal field.

When the value-translation-type attribute is specified, you must also specify a `<value-translation>` sub-element that defines the exact method of translating from one to another.

- `comparable`
  This attribute has no meaning to the mapping process itself. However, it is used by the Gateway Wizard to help determine the set of fields displayed in the differencing and changes preview before starting the import process. Valid values include:
  - `true`—Include the field in the differencing and changes preview.
  - `false`—Do not include the field in the differencing and changes preview.

  If you do not specify a `comparable` attribute, `false` is the default behavior.
  - `Parent Element`: `<mapping>, <map>`
  - `Sub-Elements`: `<default>, <value-translation>`

`<default>`

The `<default>` element specifies a value that is assigned to a field in the following situations:

- Only an external or internal field name is named
- The internal or external field from which it must be updated does not exist

The following example sets the `Category` field in the internal data to a value of `Heading` when the internal data items are initially created.

```
<field internal="Category" direction="in" on-create-only="true">
    <default>Heading</default>
</field>
```

- `Attributes`: None
- `Parent Element`: `<field>`
- `Sub-Elements`: None

`<value-translation>`

When a `<field>` element contains a `value-translation-type` attribute, the `<value-translation>` element defines which values from the external data source correspond to values in the internal data.

The following example shows how to define the correspondence between ranges of integers in the external field and a single string value in the internal field:

```
<field external="Priority" internal="Priority" direction="both"
value-translation-type="intrange-string">
    <value-translation external="0..250" internal="Low" />
    <value-translation external="251..500" internal="Medium" />
    <value-translation external="501..750" internal="High" />
    <value-translation external="751..1000" internal="Critical" />
</field>
```

- `Attributes`
  - `external`
    Specifies which value or range in the external field corresponds to the value or range in the internal field as defined by the `internal` attribute. The `value-translation-type` attribute of the parent `<field>` element specifies the type of value or range.
  - `internal`
    Specifies which value or range in the internal field corresponds to the value or range in the external field as defined by the `external` attribute. The `value-translation-type` attribute of the parent `<field>` element specifies the type of value or range.
- `Parent Element`: `<field>`
- `Sub-Elements`: None

`<map-conditional>`

The `<map-conditional>` element specifies that the mapping of a field is conditional on the value of a field or its properties. Based on the value of the field or property, a set of specialized mapping rules defined by a `<map>` element is activated.

In its simplest form, the value of the specified field or property is used as the name of the `<map>` to activate. The following example examines the value of the prototype property and searches for a `<map>` element with a name that matches the value of prototype. If located, that `<map>` is activated and the specialized mapping rules defined within that `<map>` are applied.

```
<map-conditional property="prototype" />
```

Using this example, if the value of the `prototype` property is `CONTENT` and a `<map name="CONTENT">` element exists, that `<map>` is activated.

Once a `<map-conditional>` element activates a `<map>`, all other `<map-conditional>` elements at that level are ignored. The following example first searches for a `<map>` that contains a name that matches the value of the external field name `Description`. If such a `<map>` is found, it is activated and the second `<map-conditional>` is ignored. However, if no such `<map>` is found, the second `<map-conditional>` is examined to determine if a `<map>` matches the name of the `prototype` property.

```
<map-conditional external="Description" />
<map-conditional property="prototype" />
```

If the field or property containing a value that is being compared to `<map>` names does not exist in the current item, the `<map-conditional>` is ignored.

- `Attributes`
  - `external`
    Specifies the name of a field in the external data source whose value is tested to find a matching`<map>` element.
  - `internal`
    Specifies the name of a field in the internal data whose value is tested to find a matching `<map>` element.
  - `property`
    Specifies an item control property whose value is tested to find a matching `<map>` element.
  - `direction`
    Specifies that a matching `<map>` is activated only if the direction of the relationship between the external field and internal data field matches the specified value. Possible values include:
    - `in`—The internal data field is updated from the external data field.
    - `out`—The external data field is updated from the internal data field.
    - `both`—The external and internal data fields are updated by the other, depending on context.

    The following example only activates a matching `<map>` for the `prototype` value if the internal field is updated from the external field. It is activated only if the mapping is an import operation.

    ```
    <map-conditional property="prototype" direction="in">
    ```
  - `default-map`
    Specifies the name, as defined by a `<map>` element, of a set of specialized mapping rules to apply when the value matched to `<map>` names does not match any `<map>` elements and does not contain any other recognized value. This means that it is a null value or its value is not represented in any `<map-translation>` sub-element.
    - `Parent Element: <mapping>, <map>`

- `Sub-Elements:` `<map-translation>`

<map-translation>

The `<map-translation>` element is a sub-element of the `<map-conditional>` element. It lets you specify a list of values that do not match a `<map>` name) for the field or property examined in the `<map-conditional>`. It then associate them with a `<map>` element. The specified `<map>` is activated if the examined field or property contains the given value.

For the following example, if the value of the external field named `Category` is `Application`, then `<map-name="Application">` is activated. If the value is not `Application`, the `<map-translation>` sub-elements are used. If the value of the `Category` field is either `Software` or `Utility`, then the `<map-name="Application">` is also activated.

```
<map-conditional external="Category">
   <map-translation external="Software" map-name="Application" />
   <map-translation external="Utility" map-name="Application" />
</map-conditional>
<map name="Application">
...
</map>
```

Assume that no external attribute is specified in a `<map-translation>` sub-element. Any non-null value for the field or property specified in the parent `<map-conditional>` element that has not already been matched activates the specified `<map>`. For the following example, if the value of the `Category` field is `Application` or `Non-Application`, the `<map>` element with that name is activated. If the field contains another value, the `<map-translation>` sub-elements specify that a value of `Software` or `Utility` for that field activates `<map-name="Application">`. Any other non-null value activates `<map-name="Non-Application">`.

```
<map-conditional external="Category">
   <map-translation external="Software" map-name="Application" />
   <map-translation external="Utility" map-name="Application" />
   <map-translation map-name="Non-Application" />
</map-conditional>
 <map name="Application">
<map name="Application">
...
</map>
<map name="Non-Application">
...
</map>
```

You can only specify one `<map-translation>` sub-element without an external attribute per `<map-conditional>` element.

- `Attributes`
  - `external`
    In a `<map-translation>` element, the `external` attribute does not refer to the name of a field in the external data source. Rather, it refers to a possible value for the field or property in the parent `<map-conditional>` element that is matched to `<map>` names. In the examples, the external attributes in the `<map-translation>` elements specify possible values for the external field named `Category`.
  - `map-name`
    Specifies the name, as defined by a `<map>` element, of a set of specialized mapping rules to activate when the field or property matched in the parent `<map-conditional>` contain a value that matches the value specified by the `external` attribute.
    Assume that no external attribute is specified and that the field or property matched contains a non-null value. The specified `<map-name>` is activated, regardless of the value of that field or property. Only one `<map-translation>` sub-element per `<map-conditional>` is allowed.
- `Parent Element`: `<map-conditional>`
- `Sub-Elements`: None

<map>

The `<map>` element defines a specialized set of mappings (using `<field>` elements) that are only performed when activated by a `<map-conditional>` element. When a `<map>` element is not activated, you can nest `<map>` elements, allowing you to specialize the mappings to be performed even further.

In the following example, the `prototype` property of the item is examined. The main level of the Gateway Configuration file is the level where `<map-conditional property="prototype"/>` resides. This level is searched for a `<map>` with a name that matches the value of the `prototype` property. If `prototype` contains a value of `CONTENT`, the `<map name="CONTENT">` element is activated.

```
<map-conditional property="prototype"/>
...
<map name="CONTENT">
   <map-conditional external="Category"/>
   ...
   <map name="Software Requirements">
   ...
   </map>
   <map name="Hardware Requirements">
   ...
   </map>
</map>
```

Within this `<map>`, the `<map-conditional external="Category">` element examines the value of the external data source field named `Category`. This field identifies a category of requirement and looks to activate a `<map>` with a name matching the value of `Category`. Such a matching `<map>` must exist at the same level as that `<map-conditional>` within the `<map>` element. In this case, `<map name="Software Requirements">` or `<map name= "Hardware Requirements">` is activated if the `Category` field contains the value `"Software Requirements"` or `"Hardware Requirements"`, respectively.

By using nested `<map>` elements, you can create complex AND/OR structures for your conditionals. For example, the previous example represents the following logic:

- If the prototype is `"CONTENT"` AND the `Category` field is `"Software Requirements"`, use the `Software Requirements` mappings.
- If the prototype is `"CONTENT"` AND the `Category` field is `"Hardware Requirements"`, use the `Hardware Requirements` mappings.
- `Attributes`
  - `name`
    Specifies a name for this particular set of mapping rules. This is the name specified with the `default-map` attribute of the `<map-conditional>` element or the `map-name` attribute of the `<map-translational>` element.
- `Parent Element`: `<mapping>, <map>`
- `Sub-Elements`: `<set-property>, <link-field>, <field>, <map-conditional>, <map>`

Custom Field Values

Mapping Custom Field Values

There are two ways to map Custom Field values:

- Individual custom field values

  A user can map individual custom field values as follows:

  ```
  <field external="Vendor Name"
  direction="both"
  internal="Vendor Name"
  field-type="custom-field-value"/>
  ```

A new field-type attribute value "custom-field-value" is introduced to support individual custom field value attribute.

To map individual custom field values the following property needs to be set when the mapping configuration is configured:

```
<set-property name="custom-field-values" value="Custom Field Values">
```

The value attribute represents the name of the “custom-field-values” field in Integrity Lifecycle Manager.

Note

You need to specify the

```
<set-property name="custom-field-values" value="Custom Field Values">
```

property for the individual custom field value mapping to work

This property indicates the name of field in Integrity Lifecycle Manager having `customfieldvalues` data type. The value must be specified when an individual custom field value mapping is used. This property is used to identify that the custom field value mapping is individual.

For a custom field value of date type the `data-type="date"` attribute is used. `external-date-format` or `internal-date-format` attributes are meaningful only if `data-type=date` attribute is specified. For more information, see the Integrity Lifecycle Manager Help Center.

- Complete custom field values

  User can map complete custom field values

  ```
  <field internal="Custom Field Values"
  direction="both"
  external="All Imported Attributes/>
  ```

  For this mapping to be used verify that the `custom-field-values` property is not specified in the mapping (that is, the element`<set-property name="custom-field-values" value="Custom Field Values">` should be absent).

Caution

In this type of mapping the imported values replace the existing specified values. The unspecified values are removed. This works similar to the `--field="`*`Custom Field Values`* option.

For more information, see the Integrity Lifecycle Manager Help Center or the *Integrity Lifecycle Manager Installation and Upgrading Guide*.

Edit in Word

Mapping the individual custom field values in the Edit in Word command is not currently supported. However, a user can map complete custom field values field in the .dotx template file. To edit the values it is mandatory for the user to specify all the custom field values to use this mapping.

- For an overview of exporting Integrity Lifecycle Manager documents, see Gateway Export Overview on page 9.
- For details on exporting, see Exporting Content on page 29.
- For an overview of importing Integrity Lifecycle Manager documents, see Gateway Import Overview on page 8.
- For details on importing, see Importing Content on page 17.

Sample Gateway Configuration File

The following sample Gateway Configuration file shows how various XML elements described in the previous section work together:

```
<?xml version="1.0"?>
<mapping name="IIF_Import_Sample_Template" template-version="2.1">
    <description>
    Sample Generic template to import/export IIF documents
    New documents are imported as an MKS RM "Input Document".
    </description>

    <set-property name="owner" value="external"/>
    <set-property name="prior-data-field" value="Shared Attachments"/>
    <set-property name="prior-data-name" value="source-document.iif"/>

    <!-- Internal MKS RM 2007 defaults when importing a new document -->
    <field internal="Type"
       direction="in"
       field-type="type"
       field-type="type"
       <default>Input Document</default>
</field>
<field external="MKS_INPUT_PROJECT"
     internal="Project"
     direction="in"
     on-create-only="true">
     <default>/Release2</default>
</field>
```

```
<!-- Specific mappings defined below. -->

<map-conditional property="prototype" />

<map name="DOCUMENT">

<!-- Internal MKS RM 2007 defaults when importing a new document -->

    <link-field external="MKS_DOCUMENT" field-type="id" />

    <field internal="Shared Category"
     direction="in"
     n-create-only="true">
     <default>Document</default>
</field>

<field external="DOCUMENT_TITLE" internal="Summary" required="true" />

<field external="Content"
     internal="Shared Text"
     data-type="xhtml"
     field-type="richcontent"
    attachment="Shared Attachments" />

<!- When the Section column is visible, and a DOCX template that
contains section template elements is specified, section values
are included in theexport -->

<field external="Section"
     internal="Section"
     direction="Out" />

    <!-- Additional metadata generated by the parser but currently
 unused-->
    <field external="DOCUMENT_ID" />
    <field external="INSTANCE_ID" />

</map>

<map name="CONTENT">

    <!-- Within a PDF, we have the possibility that some content
    (but not all content) may be tagged with a REQUIREMENT_ID
    that can be used to more accurately associate the content
    with content previously imported. Thus we use it as an
    *optional* link field.
```

```
-->

    <link-field external="REQUIREMENT_ID" internal="ImportedReqID"
required="false" comparable="true" />

    <!-- The MKS field that is to store the external requirement
     ID is expected to be a short-text (string) field. On some
     database products, such as Oracle, performing a lookup
     based on such a string field is non-optimal. To aid in the
     lookup, you can optionally also store a hash value for the
     requirement ID as an integer field in MKS. To do so, add
     the following attribute to the link-field above ...

     hash-code="ImportedReqID_hash"

     NOTE: As above, "ImportedReqID_hash" (or something
      similar) must be created as an integer field in MKS and
      made visible on the Requirement Content Item type(s).
-->

<field external="Content"
     internal="Text"
     data-type="xhtml"
     field-type="richcontent"
     attachment="Text Attachments" comparable="true" />

<!-- By default, content is categorized as an imported
   requirement -->
<field internal="Category">
    <default>Imported Requirement</default>
</field>

<!-- Some specialized content may be re-categorized as
    headings -->
<map-conditional external="CHAPTER_HEADING">
    <!-- If we have a value for CHAPTER_HEADING, treat it as a
      Heading -->
    <map-translation map-name="Heading" />
</map-conditional>
<map-conditional external="SECTION_NUMBER">
    <!-- If we have a value for SECTION_NUMBER, treat it as a
      Heading -->
    <map-translation map-name="Heading" /><map-translation
 map-name="Heading" />
</map-conditional>
```

```
<!-- more generically (remove if this is too generic) ... -->
<map-conditional property="children">
    <!-- If the content has any children then we treat it as a
      Heading -->
<map-translation map-name="Heading" />
</map-conditional>

<map name="Heading">
    <field internal="Category">
      <default>Heading</default>
    </field>
  </map>

  </map>
</mapping>
```

7

Microsoft Word Support

This section provides details on specific support for exporting Integrity Lifecycle Manager documents to the Microsoft Word DOCX format.

- For details on DOCX format, see http://msdn.microsoft.com/en-us/library/aa338205.aspx.
- For details on supported versions of Microsoft Word, see http://www.ptc.com/support/integrity.htm.

Replacing the Microsoft Word Integration with Gateway

If you currently use the Microsoft Word integration, this section contains information to help you move from it to using Gateway for importing and exporting Word documents. Before moving from the Word integration to using Gateway, consult PTC - Integrity Lifecycle Manager Support for assistance.

The following information is intended to provide insights into the process. This information is not intended for use as a complete set of tasks. You might need to take additional considerations into account before replacing the Word integration with Gateway.

- The XSLT processing was moved from the Microsoft .NET Framework model to Java's XALAN model. The XSLT namespace definition changes include:

  ```
  xmlns:mks="http://www.mks.com/schemas/MKSItems"
  xmlns:id="mks://com.mks.gateway.driver.XSLTIntegrationDriver"
  xmlns:local-variables="mks://com.mks.gateway.driver.XSLTIntegrationDriver"
  xmlns:stack="mks://com.mks.gateway.importer.word.ResourceStack"
  xmlns:level="mks://com.mks.gateway.importer.word.HeadingLevelStack"
  ```

- The DSD is expected to generate output that is consistent with the Gateway Item Interchange Format (IIF) model. Some changes include:
  - The `mks:items` element becomes `mks:MKSItems`.
  - The `mks:item` element becomes `mks:Item`.
  - The `mks:field` element becomes `mks:Field`.
  - The `prototype` attribute needs to be added to `mks.Item`.
  - The `type` attribute on `mks.Item` becomes a field.
  - Rich content is represented under the `mks:Richcontent` element and must be surrounded with the `<div> </div>` XHTML element.
  - There is a new `mks:Children` element. This addition can result in a significant change to the XSLT file. To aid in the conversion, the Gateway DSD transformer uses the `local-variables: addAsChild` method to auto-generate the `mks:Children` structure. This is similar to what the Word integration did.
- There is an updated mapping template mechanism. As a result, the `attachmentField (MKSAttachment)` no longer needs to be outputted for embedded content. If attachments are still needed, they need to be outputted under the `mks:Attachments` element as per the Gateway IIF model.
- The condensing and outline level sample configurations included with the Integrity Lifecycle Manager client can serve as examples on how to convert your XSLT files.

Exporting to DOCX Format

Gateway requires a DOCX template to be used for export. For more information, see DOCX Template on page 89.

- For an overview of exporting Integrity Lifecycle Manager documents, see Gateway Export Overview on page 9.
- For details on exporting, see Exporting Content on page 29.

To export to DOCX format, select **Microsoft Word Exporter** from the **Export Configuration** list in the Gateway Export Wizard. This list is configurable. For more information, see Exporter Configuration on page 96.

The following sample configurations are available:

- **Microsoft Word Reimport Exporter**

  Use this configuration when you need to reimport the document later. It adds hidden text containing the internal Integrity Lifecycle Manager ID to identify the document on reimport. To ensure that the ID moves with the rest of the content, carefully perform cut and paste operations.
- **Sample Microsoft Word Exporter – simple headers and content**

  This configuration outputs the document as a simple Microsoft Word document with headings.
- **Sample Microsoft Word Exporter – with TOC page**

  This configuration adds a table of contents page, the values for which can be updated in Word.
- **Sample Microsoft Word Exporter – with columns**

  This configuration outputs the document in column format.
- **Sample Microsoft Word Exporter – with tables**

  This configuration outputs the document in tabular format.
- **Sample Microsoft Word Exporter – with tables and Xpath**

  This configuration outputs the document in tabular format. It uses Xpaths to determine if rows are top rows or later rows and only outputs heading rows where needed. Other choices are also made using Xpaths.

Key Considerations

Before exporting to DOCX format, consider the following:

- To support tables and figures created in Integrity Lifecycle Manager but not imported from the DOCX format, you must configure properties in the Gateway Configuration file with the `template` property used for the export. Configure these properties:

  ```
  <property name="figureCaptionLabel">figure</property>
  ```

where `figure` is the figure caption label
`<property name="tableCaptionLabel">table</property>`
where `table` is the table caption label

For more information on configuring `<property>`, see Export Configuration on page 57.

- For table of contents support, the document must use valid TOC fields.
- After exporting the document, all Table of Figures and Table of Contents fields must be updated.
- The captions for the figures and tables are numbered based on the rich content field data. If the template also contains figures and tables with the same captions, the numbering is not correct. In this case, the caption fields must be updated before updating the Table of Figures.
- The figure description (when not imported from DOCX) is based on the `alt` attribute value of `<img>`.
- The table description (when not imported from DOCX) is based on the value of `<caption>`.
- Empty paragraphs are removed during export (if the empty paragraph is in a table cell, it is retained). To retain these paragraphs, insert a placeholder (such as a space) before exporting.
- Do not use `_Toc*` in the bookmark name. Such bookmarks are not included when the document is exported to DOCX format. Instead, create a new bookmark to the figure, heading, or table that you want to reference. Editing the bookmark is also not advised because it prevents the link in the table of contents from working. As a best practice, create the reference in the DOCX file before import. This creates `_Ref*` style bookmarks that are imported and exported without problems.
- During export of Integrity Lifecycle Manager documents to DOCX format, Gateway supports the following rich text features:
  - Fonts and font sizes, except for the following:
    - Font size in fractions is not supported by Gateway. For example, when a MS Word document with font size 10.5 is exported using Gateway Export Wizard, the font size is changed to 10 when opened in MS Word.
    - Font size 11 in a MS Word document is not supported by Gateway. The font size is changed to 10, which is the default font size in Gateway.
  - Text formatted as superscript, subscript and justified.
  - Special characters available through the rich content toolbar in the Integrity Lifecycle Manager client GUI and also all supported HTML entities.

- Mathematical formula is exported to Microsoft Word document without loss of data. In rich text field, {√n}² is displayed instead of mathematical formula.
- On export, DOCX file default fonts replace fonts that are not available in the operating system with no loss of content.
- While exporting to DOCX, format for date is decided on the following attributes defined in the configuration file:
  - When `field-type` is specified as `date` or `date-time`, and `external-date-format` or `external-locale` is not defined, Integrity Lifecycle Manager client's default format for date is selected. In this case, MS Word uses format `MM/DD/YYYY` for `date` field and format `MM/DD/YYYY hh:mm:ss a` for `date-time` field.
  - Format for date specified in the DOCX template file overrides the format specified in the configuration file.
  - When none of the attributes from `field-type`, `external-date-format` or `external-locale` are specified, MS Word uses the format for date and time as per the Integrity Lifecycle Manager client's system locale.

DOCX Template

The DOCX template is a DOCX file used by Gateway to determine the formatting for the exported document. This template contains all field names and their formatting tags. For more information, see Template Elements on page 90.

Key Considerations

When creating or modifying a DOCX template, consider the following:

- Use straight quotation marks (") only. Smart quotation marks (”) are unsupported in the template.
- During export, rich content from an Integrity Lifecycle Manager field overrides the specified template format, depending on the attribute indicating whether to apply rich text.

  If the `richcontent` attribute is set to `false`, then rich content is converted to plain text and formatted using the template. When a style is applied in this manner, images, hyperlinks, horizontal lines, and tables are not supported for this field on export.

  Similarly all rich content fields used in footnotes and endnotes are converted to plain text.

Exported documents containing lists that have a style defined in the template must have the `List Paragraph` style applied for them to be reimported correctly.

- Merged table cells are supported.
- The file extension for the template file must be DOCX.
- Spaces after <% and before %> portions of the tags are ignored in all cases. A field name must be wrapped in brackets of type: { }. Quotation marks (") around fields are not required, but they are permitted.
- Attachments, including in-line image attachments and locally linked attachments, are exported to an attachment directory at the root directory to which the DOCX file is exported. The attachment directory uses the same name as the DOCX file, with `- MKS_Attachments` appended. If an attachment file is referenced but does not exist, a warning message is outputted to the log file. For more information, see Gateway Logging on page 33.
- When exporting, empty paragraphs, which are lines containing no text, are not present in the exported DOCX file. To create a blank line separating text, insert a non-visible character such as a space or tab. This ensures that the line is no longer an empty paragraph. Alternatively, create a blank line by pressing SHIFT + ENTER to insert a soft line return.
- Images and tables that are larger than the width for display in the exported file are proportionately shrunk to fit.
- Exported subdocuments do not have a section number. Items under the subdocument have their section number reset.
- When exporting, node field values are not retained if the node points to a subsegment, such as `Priority` field value.
- Template elements used in the header and footer are limited to fields of the document root item. Content item fields cannot be specified.
- Exporting to a template that includes text boxes has limited rendering capabilities. For example, some text can overlap for non-rich content fields. For best results, use rich content fields in Integrity Lifecycle Manager.

Template Elements

The following elements are used to construct a DOCX template.

Field Name

Contents replaced with specified field name contents. The `key="value"` pairs denote valid attributes for the field. This tag can be used in the document properties of the DOCX file. For example, you can use it for `Title`, `Subject`, and `Author`. However, these fields must be mapped to fields in the root document.

Start tag:

```
<%{field name}
key1="value1"
key2="value2"%>
```

End tag: None

Comments

Denotes commented text, which has no impact on the export output. Use this tag to comment elements in your template.

Start tag:

```
<%// comment ...%>
```

End tag: None

Content

Specifies the begin and end of a format block for content. Each template must include these tags once.

Start tag:

```
<%beginContent%>
```

End tag:

```
<%endContent%>
```

Level

Specifies the begin and end of a format block for a content level as per the document hierarchy. If specified without `levelNumber`, the formatting specified for the tag becomes the default format used for all levels that do not have formatting tags specified.

Start tag:

```
<%beginLevel
levelNumber%>
```

End tag:

```
<%endLevel%>
```

Conditional String

Specifies a conditional format block, where the format is only applied if the conditions are met. Conditions are only checked as part of the content-level block.

Start tag:

```
<%if condition_string1%>
<%elseif
condition_string2%>
<%else%>
<%endif%>
```

End tag:

```
<%endif%>
```

Timestamp

Specifies to include a timestamp based on the data and time in the exported file. The date and time provided are from the Integrity Lifecycle Manager server database. They are not from the client machine on which Gateway is installed.

Start tag:

```
<%{attribute:value}%>
```

End tag: None

Value: `asof` or `now`

- `asof`

  Specifies to include a timestamp in the exported document as of the date and time selected in the wizard. You can also specify the date format. An example follows:

  ```
  <% {attribute:asof}
  dateFormat= "MM/dd/yyyy
   HH:mm:ss z" %>
  ```

- `now`

  Specifies to include a timestamp in the exported document with the current date and time from the Integrity Lifecycle Manager server database. You can also specify the date format. An example follows:

  ```
  <% {attribute:now} dateFormat=
  "MM/dd/yyyy HH:mm:ss z" %>
  ```

For more information on date formats, see the man pages for the command line interface.

Element Syntax and Semantics

The following are guidelines to follow when using elements in the DOCX template:

- Only a limited number of elements are needed. For more information, see Template Elements on page 90. They share the format of `<%keyword parameters%>`. All elements must start with "<%" and end with `"%>"`. Spaces are permitted after `"<%"` and before `"%>"`.
- Surround a field name with `"{" and "}"`, such as `<% {fieldname} %>`. The value of the field represented by the field name is used to substitute the element.
- The comment element has no impact on the output. It is in the form `<%// comment ...%>`. The comment element can be placed anywhere in the template.
- When using a condition string, consider the following
    - Use {} to identify field names.
    - Use () to identify groups, including individual clauses separated by boolean operators, such as "and" and "or". For example: `(({fieldName1} == “value1”) or ({fieldName2} > “value2”)) and ({fieldName3} <= “value3”).`
    - Use lowercase “and” and “or” for AND and OR.
    - Valid operators are limited to ==, !=, >, >=, <, and <=.
    - The predefined value must be a string that is specified between straight quotes (" ").
- There is a strict hierarchy for elements used in the template. There must be one content block. Use level blocks and condition blocks only within a content block.

  A level block can contain a condition block, but a condition block cannot contain a level block. An example of a level block follows:

```
<%beginContent%>
  <%beginLevel 1%>
    <%if ...%>
    ...
    <%endif%>
  <%endLevel%>
  <%if ... %>
  ...
  <%endif%>
  <%beginLevel 2%>
  ...
  <%endLevel%>
  <%beginLevel%><%// this is the default level%>
```

```
      ...
      <%endLevel%>
    <%endContent%>
```

Valid Field Attributes for DOCX Templates

An attribute is specified through a `key=value` pair, where the key is alphanumeric and the value is encased in double quotation marks.

The following characters must be escaped with a backslash (\) for the field values:

- equal (=)
- double quotations marks (")
- comma (,)
- backslash (\)

The following are field attribute valid keys:

- `dateFormat`—Specifies a date format for the field. The format must be a Java supported date format.
- `richcontent`—Used for rich content fields. The field is exported as rich content if the attribute is set to the value `true`, which is the default value. If the value is `false`, the field is exported as plain text.
- `hyperlink`—Specifies to represent the field as a hyperlink in the exported document. To generate a hyperlink, the attribute requires a string value of `true`. The key benefit of using this attribute is hyperlinks are dynamically generated for the document. The hyperlinks use the server and hostname information that Gateway is using to connect to the Integrity Lifecycle Manager server when exporting the document. This makes the hyperlinks accurate and relevant to users on this system.

  For example, when specified with the ID field, users of the Integrity Lifecycle Manager Web interface can click the ID to view the item.

  To use the `hyperlink` attribute, an appropriate field type must be specified in the DOCX template. You can use an ID or relationship field. The `<link field>` element must also be specified for this field in the Gateway Wizard Configuration file.

  By default, hyperlinks are in this form:

  `http://hostname:port/im/viewitem?selection=itemID`
  where:

  - `hostname` is the Integrity Lifecycle Manager server hostname.
  - `port` is the Integrity Lifecycle Manager server port number.
  - `itemID` is the value for the Integrity Lifecycle Manager item ID number.

Note

Assume that you do not want Gateway to dynamically generate the server and port. In the Gateway Wizard Configuration file (`gateway-toolconfiguration.xml`), you can specify a server and port using the hostname and port properties in the desired configuration block. That server and port information is then used for all hyperlinks created during export using the `hyperlink` attribute. For an SSL enabled server, set the `isSSL` property to `true` to generate the hyperlinks using the `https` protocol.

When using the `hyperlink` attribute for relationship fields, you must specify the field using the field type `relationship` in the Gateway Configuration file. Each item ID in the resulting list then has its own hyperlink. List items are comma delimited.

Note

For backwards compatibility, hyperlinks specified in the DOCX template that you code manually are still used by Gateway during export. The `hyperlink` attribute does not work if a hyperlink is also specified in the DOCX template for the same field instance.

XML Path Language Support

Gateway supports the XML Path (XPath) language, which provides a way to access any part of the document from any part of the template. For information on the XPath language, see http://www.w3.org/TR/xpath.

Some uses for XPaths in the template include returning counts of descendants, successors, predecessors, or siblings. A few examples follow:

- `<%{xpath:count(./descendant::*)}%>` returns the number of descendants for the document.
- `<%{xpath:count(./preceding-sibling::*)%>` returns the number of preceding siblings.

When referencing items and fields, items are element objects. Fields and their values are treated as attributes on the item elements. You can use XPath to reference a field value in the preceding sibling item like this:
`./preceding-sibling::*[1]/Field[@name='FieldName']/text()`

However, it is preferable to use this simpler format:
`./preceding-sibling::*[1]/@FieldName`

Any field referenced using the simpler format cannot contain spaces in its name. XML, and therefore XPath, do not support spaces in attribute names. Assume that a field on the Integrity Lifecycle Manager server contains a space. You use the mapping template to remove spaces from the field name for use with XPath statements in DOCX templates.

Sample DOCX Templates

Sample DOCX template files are available in the following location:

`installdir/config/gateway/export/Microsoft Word/`
where *installdir* is the installation location of the Integrity Lifecycle Manager client.

The included template files are samples intended for use with commonly implemented solution configurations:

- `2009-WordExportTemplate-01-Tables.docx`—Sample template elements for tables
- `2009-WordExportTemplate-02-Tables-Xpaths.docx`—Sample template elements for tables and XPath support
- `2009-WordExportTemplate-03-Columns.docx`—Sample template elements for columns
- `2009-WordExportTemplate-04-TOC.docx`—Sample template elements for a document table of contents
- `2009-WordExportTemplate-05-Simple.docx`—Sample template elements for headers and content.
- `2009-WordExportTemplate-06-Reimport.docx`—Sample template elements for use with the provided reimport configurations

If necessary, you can modify these template files for use with your implementation of Integrity Lifecycle Manager.

Exporter Configuration

PTC provides default configurations for exporting to DOCX format. This topic provides information on configuring Gateway to meet your needs. If necessary, you can modify the Wizard Configuration file. For details on configuring specific XML elements in this file, see Export Configuration on page 57.

The following XML elements are provided by default for exporting to DOCX format:

- `<name>`—Name displayed in **Export Configuration** list in Gateway Export Wizard.
- `<description>`—Description displayed in **Export Configuration** list in Gateway Export Wizard.

- `<exporter id>`—Property sent to the exporter, such as `MSWORD`.
- `<property name="template">`—Location of the DOCX file template used for the export. An example follows:

  ```
  http://hostname:port/test.docx
  ```

  where:

  - `hostname` is the server hosting the document.
  - `port` is its port number.

  Location can be configured the following ways as per the URL syntax:

  - `http:// server location`
  - `https:// server location`
  - `file:/// file location`
  - `file local file location`
  - `/ file location`
  - `../ relative local file location`

  Note

  Specify server paths relative to the following location: `*installdir*/data/public_html/`.

- `<extension>`—File extension for the output file, such as`docx`.
- `<display-fields>`—Field used by the DOCX template as the document title.
- `<gateway-configuration-name>`—Name of a Gateway configuration. For more information, see Gateway Configuration File on page 64.

Importing DOCX Files

For an overview of importing Integrity Lifecycle Manager documents, see Gateway Import Overview on page 8. For details on importing, see Importing Content on page 17.

Note

If you are importing a DOCX file that contains ordered lists, see List Support on page 114 for important information that you must know.

To import to a DOCX file, on the **Select Configuration** panel of the Gateway Import Wizard, select one of the following from the list of available import configurations:

- **Condensing Word Import and Re-Import**

  This configuration condenses paragraphs together into single pieces of content separated by header items. In the original Microsoft Word document, heading styles identify header items. You can use this configuration for imports and reimports. However, when reimporting, the document must be one that was exported using the reimport export template, `2009-WordExportTemplate-06-Reimport.docx`. This export template injects the PTC internal IDs into the document text at appropriate locations. Requirement documents are created with Requirement content. Items found lacking the ID are assumed to be new content and are added by the import. Exporting the document with the new items lets you reimport them afterward.

- **External ID Import and Re-Import**

  This configuration imports documents containing external IDs. This allows for the document to be reimported later without exporting it from Integrity Lifecycle Manager using the reimport export template, `2009-WordExportTemplate-06-Reimport.docx`. The external IDs are necessary in this case so that Gateway can identify the document as one it has processed previously. The external IDs must be unique within the document but can be reused in the system in other documents.

  To use this example, the following field modifications are likely needed:

  - The **External ID** field must be added to Requirements documents.
  - The **Requirements**, **Risk**, and **Priority** fields must be added to requirements.
  - The start of each content must contain `External ID:Risk:Priority` if you want to identify these fields on reimport. The text can be hidden so that it does not bother document readers. However, editors must take care to ensure that the IDs are copied when cut and paste operations are performed.
  - The document ID must be supplied in the Gateway Import Wizard if you reimport. An example can be extended by adding an external document ID. An attempt is made to recognize heading items lacking external IDs by content. If the headings match exactly, they are maintained. If they do not match exactly, delete and add operations are performed.

- **Outline Levels Condensing Import with Level 2 Sub-Documents**

  This configuration creates a condensed document like the **Condensing Word Import and Re-Import** configuration. However, it also creates subdocuments whenever level 2 content is found. These subdocuments are created as referenced subdocuments within Integrity Lifecycle Manager. Subdocuments cannot be reimported currently. Instead of subdocuments, different content categories can be created at different levels by extending this example.

- **Requirements Document Import and Re-import with Outline Levels**

  This configuration parses a requirements document into sections according to the outline level of the original document. Sequential paragraphs in the same section are mapped into separate requirements in the destination requirements document.

- **Requirements Document Outline Levels Condensing Import**

  This configuration parses a requirement document into sections according to the outline level of the original document. Sequential paragraphs in the same section are condensed into the same requirement in the destination document.

- **Test Suite Condensing Word Import and Re-Import**

  This configuration is similar to the **Condensing Word Import and Re-Import**configuration. However, it creates test suites containing test cases. If the document is subsequently exported using the reimport export template, `2009-WordExportTemplate-06-Reimport.docx`, it can be reimported again. This configuration condenses multiple paragraphs into single test cases.

- **Test Suite Import and Re-import with Outline Levels**

  This configuration parses a test suite document into sections according to the outline level of the original document. Sequential paragraphs in the same section are mapped into separate test cases in the destination test plan.

  The configuration is similar to the previous one, but it is not intended for reimports. It also does not condense the content into single test cases.

To modify what configurations display in the list, as well as the parser used, see Importer Configuration on page 103.

PTC provides a sample parser, `MSWord-DSD`. If necessary, you can modify this parser to meet your needs. For information on writing a DSD, see Writing a DSD For DOCX Import on page 104.

Key Considerations for Importing DOCX Files

Before importing a DOCX file, consider the following:

- To import a DOCX file that contains OLE objects, you must have the corresponding Microsoft application for that object installed on your system. For more information on support for embedded objects, see Embedded Object Support on page 108. Some scenarios follow for clarifying when Microsoft application must be installed on your system.

    - If the DOCX file contains only paragraphs of text and no OLE objects, Microsoft Word or Office does not need to be installed.
    - If the DOCX file contains embedded images or other OLE objects not created by Microsoft Word or Office, such as sounds or video clips, Microsoft Word or Office does not need to be installed. However, its installation is recommended.
    - If the DOCX file contains embedded OLE objects created using Microsoft Office 2003, Microsoft Word or Office must be installed.
    - If the DOCX file contains embedded OLE objects created in Microsoft Visio, then Microsoft Visio must be installed.
    - If the DOCX file contains an embedded Microsoft Project object, Microsoft Project must be installed.
- To prevent the creation of empty items, Gateway validates imported documents and removes empty paragraphs.
- Password-protected files are not supported for import using Gateway.
- During import, double strike formatted text is converted to single strike text. Ensure that this formatting change does not alter the meaning of your document.
- Gateway ignores numbering definitions in headings if the paragraph style participates in the outline level.
- Assume that you are reimporting a DOCX file that was exported from Integrity Lifecycle Manager and it has no usable IDs. Gateway attempts to match textual changes to existing items, even if these items were moved within the document tree. Larger requirements with changes are easier for Gateway to match than smaller requirements with changes. Formatting changes are not considered comparable differences in large requirements. If items cannot be identified, then Gateway creates new items and drops old ones as needed. Items that cannot be identified either have no ID or change that are not significant enough.
- When Gateway imports the DOCX file, it creates an image capture of embedded images with text. If you modified the default paragraph styles, the text can appear cutoff in the resulting image attachment thumbnail in Integrity Lifecycle Manager. However, no information is lost. You can edit the original attachment from Integrity Lifecycle Manager to display as desired or edit within the resulting DOCX file, if the document is exported.
- To successfully import a DOCX file that contains images inserted using **Link to File**, it is recommended that you insert the images using **Insert ▸ Object** in Microsoft Word.
- During import, colors and styles for table border and cell are not retained. The table border width defaults to `border = 1`.

- In DOCX files, the padding for table cells can be defined individually. However, Integrity Lifecycle Manager documents use the same cell padding for all cells in a table. During import, cell padding is determined by using the first non-zero value defined in the table in the order of top, left, bottom, and then right. For example, assume that the cell padding value is 0 for top, 0 for left, 2 for bottom, and 3 for right. The value 2 is used for all cells in that table in the resulting Integrity Lifecycle Manager document.
- Before import, PTC recommends that you ensure that the source document is saved and not open in Microsoft Word for editing. Microsoft Word includes automatic saving functionality that can produce undesirable results during import of the document if the source document is not closed or explicitly saved.
- During import, the background color support is as follows:
  - If content is highlighted in Microsoft Word, the highlight color is converted to the corresponding hex value for HTML in rich text.
  - If content is shaded in Microsoft Word (`<w:shd>`), only the `clear` and `solid` values in combination with a hex color value are supported. Examples follow:
    - If the attribute `w:val="solid"`, the value in attribute `w:color` is used.
    - If the attribute `w:val="clear"`, the value in attribute `w:fill` is used.
    - If the attribute `w:val` value is anything other than solid or clear, such as `pct15`, the color is ignored. If the `w:color` or `w:fill` value is `auto`, the color is also ignored.
- Due to the underlying HTML editor used by the Integrity Lifecycle Manager client, indentations based on character (ch) widths is currently unsupported. When a character-based indent is encountered during import, that indent is not displayed in the Integrity Lifecycle Manager client. To ensure that the integration recognizes and coverts indents so that they are correctly displayed in the Integrity Lifecycle Manager client, do not use character-based measurements. Instead use indents based on the unit measurements provided in Microsoft Word. These include inches, centimeters, millimeters, points, and picas. Character-based indents are correctly displayed in the Integrity Lifecycle Manager Web interface.
- Due to the limitations of HTML, only standard left indents selected as part of the paragraph format in Microsoft Word are recognized during import. Special indents, such as those for the first line and hanging indents, are not recognized or converted.
- If the source document contains shaded background colors, the color is only retained if the shading is 100%.

- Section numbers in the DOCX file are not imported. This includes case where some requirements have section numbers and some do not. The Integrity Lifecycle Manager document model calculates section numbers based on the `Contains` relationship. The Microsoft Word section number is not used on import or saved for later use on export.
- During import, paragraphs indented by both the body and indent amount are indented only by the body amount. For example, if the indent is 7 characters + 3 characters, the indent is 7 rather than 10.
- The following rich text content is imported to Integrity Lifecycle Manager from the DOCX file:
  - Fonts and font sizes, except for the following:
    - Font size in fractions is not supported by Gateway. For example, when a MS Word document with font size 10.5 is imported using Gateway Import Wizard, the font size is changed to 10.
    - Font size 11 in a MS Word document is not supported by Gateway. The font size is changed to 10, which is the default font size in Gateway.
  - Text formatted as superscript, subscript and justified.
  - Special characters available through the rich content toolbar in the Integrity Lifecycle Manager client GUI and also all supported HTML entities.
- Mathematical formula is imported to Microsoft Word document without loss of data. In rich text field, $\{\sqrt{n}\}^2$ is displayed instead of mathematical formula. A mathematical formula cannot be edited in rich text field. A user can edit it by selecting **Document ▸ Edit in Word** option from Integrity Lifecycle Manager. A user can also export the Microsoft Word document, edit the mathematical formula and import it back using Gateway Import Wizard.
- While importing a Microsoft Word document, the alignment that is set for mathematical formula is ignored and changed to inline. Hence, before importing the document, it is recommended to set the alignment to `Change to Inline`. This setting makes the formula run inline with the text while editing the document in Integrity Lifecycle Manager.

Note

Any mathematical formula that is resized in Integrity Lifecycle Manager is ignored while editing the document in Microsoft Word or exporting it to Microsoft Word. The original size of the formula is considered instead of the changed size.

- During import, center or right alignment in tables is not retained. All tables are aligned left after import.
- Gateway does not support distributed paragraph alignment when importing DOCX files. During import, the distributed paragraph alignment is ignored.
- Properties defined in the default paragraph style definition are not used during import. For example, assume that the style for alignment is defined in the default paragraph style and that the current paragraph uses the default paragraph style. The style for alignment is not used during import.

Importer Configuration

PTC provides sample configurations for importing a DOCX file into Integrity Lifecycle Manager. This section provides information on configuring Gateway to meet your needs.

Some configuration of the Wizard Configuration file could be necessary. However, the following XML elements are provided for importing DOCX formatted files:

- `<name>`—Name displayed in the **Import Configuration** list in the Gateway Import Wizard.
- `<description>`—Description displayed in the **Import Configuration** list in Gateway Import Wizard.
- `<parser id>`—Property sent to the importer, such as `MSWORD-DSD`.
- `<property name="xslt">`—Location of the DSD file used for the import, such as:
  `http://hostname:port/test.xslt`
  where:
  - `hostname` is the server hosting the file.
  - `port` is its port number.

  Location can be configured the following ways as per the URI syntax:
  - `http:// server location`
  - `https:// server location`
  - `file:/// file location`
  - `file local file location`
  - `/ file location`
  - `../ relative local file location`

  For information on using a DSD, see Writing a DSD For DOCX Import on page 104.

Note

Specify server paths relative to the following location: `installdir/data/public_html/`

- `<extension>`—File extension for the input file, such as `docx`.
- `<display-fields>`—Field used to populate the summary field in the root document (document title) in Integrity Lifecycle Manager.
- `<gateway-configuration-name>`—Name of an Gateway configuration. For more information, see Gateway Configuration File on page 64.

Writing a DSD for DOCX Import

To support the import of DOCX-formatted files to Integrity Lifecycle Manager, the Document Structure Definition (DSD) is an Extensible Stylesheet Language Transformations (XSLT) script. This script is used to transform an XHTML representation of the DOCX file into Item Interchange Format (IIF) items that Gateway can then process for input into Integrity Lifecycle Manager.

Note

Writing or changing a DSD requires experience in XSLT programming. Contact PTC - Integrity Lifecycle Manager Support for assistance.

Style Attributes

The following attributes are used with the paragraph `<p>` tag and heading `<h>` tags:

- `mks-word-style=internalID`, where *internalID* is an internal style ID.
- `mks-word-style-name=userVisibleName`, where *userVisibleName* is the user visible name of the style.

DSD/XSLT namespace

The following statement specifies the XSLT namespace used to reference the input XHTML data element:

xmlns:h="http://www.w3.org/1999/xhtml"

The following statement specifies the XSLT namespace used to reference and create the output IIF data elements:

xmlns:mks="http://www.mks.com/schemas/MKSItems"

The following namespaces are used to add helper functionality to the DSD (XSLT) script:

- xmlns:logger="mks:// com.mks.gateway.GatewayLogger"
  `logger:log(<messages>)` is used to create log messages from the DSD. These messages appear in the `GatewayApp.log` file.
- xmlns:id="mks:// com.mks.gateway.driver. XSLTIntegrationDriver"
  - `id:next()` is a helper function that supplies the DSD with a new (integer-based) ID. It can be used to help provide each output item with an ID.
  - `id:addAsChild(<parentID>, <childID>)` is a helper function that creates a parent-child relationship.
- xmlns:local-variables="mks:// com.mks.gateway.driver. XSLTIntegrationDriver"
  - `local-variables:setValueOf(<key>, <value>)` allows for the DSD to store (in memory) a value for a property (String) key.
  - `local-variables:valueOf(<key>)` queries the value for a property.
  - `local-variables:stringValueOf(<key>)` queries the value for a property, but returns the value in its string representation.
  - `local-variables: isResultTreeEmpty (<root>)` determines if a given tree is empty or not. It returns `true` if the given tree is empty.

Note

The property values available to the DSD include the following:

- The properties set from within the DSD.
- Properties and field values as defined by both additional parameters in the Gateway parser configuration.
- Values for user-specified fields, which you enter in the Gateway Wizard.

Any property set within the DSD is only set within the scope of the DSD itself. Once the DSD is terminated, all pre-existing properties are restored.

  - `local-variables:queue(<queue-name>, <value>)` queues a data value.
  - `local-variables:dequeue(<queue-name>)` retrieves and removes a data value from the queue.

- `local-variables:isLink(<URI-ref>)` determines if a given (String) reference is a remote hyperlink.

- xmlns:stack="mks:// com.mks.gateway.driver.word.parser. ResourceStack"
  The following are helper functions that use a general stack:
  - `stack:push(<value>)` pushes a data value onto the stack.
  - `stack:pop()` removes and returns the data value from the top of the stack.
  - `stack:empty()` returns `true` if the stack is empty.
  - `stack:peek()` accesses the data value on the top of the stack without actually removing it.

Note

There is only one stack available to the DSD.

- xmlns:level="mks:// com.mks.gateway.driver.word.parser. HeadingLevelStack"
  The following are helper functions that use a stack level:
  - `level:push(<value>)` pushes a data value onto the stack.
  - `level:pop()` removes and returns the data value from the top of the stack.
  - `level:empty()` returns true if the stack is empty.
  - `level:peek()` access the data value on the top of the stack without actually removing it.

Note

There is only one stack level available to the DSD. This is in addition to the general stack described previously.

Testing and Debugging a DSD

As stated in Writing a DSD For DOCX Import on page 104, for the purposes of its use with Gateway, a DSD is an XSLT script. If desired, the script can initially be developed using any tool or editor intended for XSLT programming. However, Gateway provides the following command:

```
gateway parse [--rawIIF] --driver=MSWORD-DSD
--param=xslt=<absolute-path-to-DSD> <absolute-path-to-Word-document>
```

This command aids in the testing of the DSD within the context of its use with Gateway and within the context of a given Word document.

The purpose of the command is to verify that the output of the DSD is valid IIF content. The command instructs Gateway to internally open the DOCX file and then convert it into XHTML.

The command then executes the DSD (XSLT) script. The script output is sent to the `gateway.log` file for review. If the script output is not valid IIF content, use the `--rawIIF` option to view the raw output from the DSD and troubleshoot the problem.

Debugging a DSD During Import

Additionally, to assist with debugging a DSD, during the import operation, you can output two files that contain the following:

- The XHTML that results from the first pass over the input DOCX document, which is what the DSD transformer operates on
- The resulting XML in IIF format after the DSD transform is complete

To output those files, specify `-Dmks.gateway.debugbasename=`*`BaseNamePath`* to `lax.nl.java.option.additional` in the following file:
`installdir``\IntegrityClient10\bin\Gateway.lax`
where:

- *BaseNamePath* is the path and base file name to which the XHTML and IIF files are written. The base file name does not include the file extension. For example, when specifying `C:/Public/debugImport`, Gateway writes `C:/Public/debugImport.xhtml` and `C:/Public/debugImport.iif`.
- *installdir* is the location where the Integrity Lifecycle Manager client is installed.

Using the Relationship Tree Adapter

Gateway adapters define how Gateway works with Integrity Lifecycle Manager. During the import process, Gateway recognizes whether to represent an imported file in Integrity Lifecycle Manager as a document for solutions that use the Integrity Lifecycle Manager document model. However, assume that your Integrity Lifecycle Manager implementation needs to represent the imported file using a relationship tree structure with a contained data set rather than the document model. You can manually configure Gateway to use the Gateway Relationship Tree Adapter.

Use the Gateway Relationship Tree Adapter when importing relationship tree-connected items as a contained data set/ For example, use it when importing a list of requirements.

The Relationship Tree Adapter: performs the following operations:

- Searches for identifier fields within the context of the relationship tree
- Automatically ensures that new items are added to the context of the relationship tree
- Maintains the relationship structure on subsequent synchronizations
- Exports item data as Gateway content items

As is the case in the Integrity Lifecycle Manager document model, the relationship tree context requires a root item to be created or specified. This root item is treated as the document item. Once enabled, the integrations provide this item ID as done when using the standard Integrity Lifecycle Manager document model.

Configuring Gateway to use the Relationship Tree Adapter requires an understanding of control properties. For more information, see `<set-property>` in Gateway XML Elements for Gateway Configuration File on page 67. Specifically, to enable the adapter, you must add the following properties to the appropriate Gateway Configuration file:

```
<set-property name="adapter" value="RelationshipTreeAdapter" / >
<set-property name="structural-relationship" value="relationshipField" />
```

where `relationshipField` is the name of the Integrity Lifecycle Manager field used to traverse the tree from its document/root item, down to its requirement items. Use the field name appropriate to your Integrity Lifecycle Manager implementation or solution.

All items are added as a child (direct or indirect) under the root item.

Embedded Object Support

When importing, Gateway supports embedded objects. The nature of the embedded object determines the following:

- How the object appears in Integrity Lifecycle Manager
- The Word editor used within Integrity Lifecycle Manager to edit documents and items in Microsoft Word
- An exported document

The following considerations apply to embedded objects:

- Word, Visio, Excel, and PowerPoint objects can be converted into standalone objects and can later be edited using the native application without requiring export, or editing the document or item in Microsoft Word. The file is stored as an in-line attachment of that file type. A PNG-formatted image displays as the representation for the in-line attachment in the document or rich content field. The objects can also be edited by editing the document or item in Word, or by exporting the document to Word.

Shapes and WordArt objects are converted to standalone Word documents and can be manipulated the same way as other standalone objects described in the preceding paragraph.

- SmartArt objects cannot be converted to standalone objects. However, they can still be edited in Word if the document or item is edited in Word or if the document is exported. The object is stored as an in-line attachment with an MKSOLE file extension. A PNG-formatted image displays as the representation for the in-line attachment in the document or rich content field.
- To ensure correct location in the Integrity Lifecycle Manager document or item's rich content field, place objects in-line with text. Using anchored objects could require you to reposition the object in Integrity Lifecycle Manager after editing.
- When importing a document with SmartArt or WordArt objects, default icons are used for these objects in Integrity Lifecycle Manager if Microsoft Word is not installed on the client machine.
- Including a large number of embedded objects in a document increases the amount of time to return to Integrity Lifecycle Manager from the Word editor or to import a document.
- Avoid using the clipboard from other applications while editing a document or item in Word or while importing a document.
- Before inserting an Excel document as an OLE object, ensure that the source document is saved and closed. Microsoft Excel includes automatic saving functionality that can produce undesirable results while inserting the document as an OLE object if the source document is not closed or explicitly saved.

Bookmark and Cross-Reference Support

Gateway supports importing bookmarks and cross-references from DOCX files. Bookmarks and cross-references are created from a source document in the following situations:

- Cross-reference to a bookmark— After import, the cross-reference appears in Integrity Lifecycle Manager as a hyperlink with the display text being referenced text.

  If a Microsoft Word document contains a cross-reference inside shapes, canvas, or group objects, after import the cross-reference appears in Integrity Lifecycle Manager as follows:

- ○ The text associated with the referenced field is retained in the corresponding shape, canvas, or group object.
- ○ The cross-reference from the shape, canvas, or group object is added below the generated thumbnail image in the same item. The text associated with the cross-reference and included in the same paragraph (`<w:p>` element) is also added below the thumbnail image.
- ○ The formatting of the cross-referenced text is retained after import.
- ○ The reference is maintained after import.

- Hyperlink to a bookmark— After import, the cross-reference appears in Integrity Lifecycle Manager as text with a hyperlink.

In the DOCX file, bookmarks can start outside of a paragraph if automatically generated. During import, the bookmark is associated with the text in the next paragraph.

For information on using bookmarks and cross-references in Integrity Lifecycle Manager, see the *Integrity Lifecycle Manager Help Center*.

How the End of a Bookmark is Determined

Gateway determines the end of a bookmark, and consequently the end of the text displayed in a cross-reference to that bookmark, in one of the following ways:

- The bookmark text ends when it is marked as ended `(w:bookmarkEnd)`.
- The bookmark text ends when the end of the paragraph is detected.
- The bookmark text ends when another bookmark is detected during import.

Best Practices When Constructing Bookmarks and Cross-References

- During import, bookmarks named `_GoBack` are not retained. Do not specify them in Integrity Lifecycle Manager because they are not retained if you export the document and then reimport it.
- When a bookmark is located in a caption paragraph, the bookmark wraps with the entire caption.
- In the DOCX file, do not overlap bookmarks with each other. During import, the start and end of book marks only results as intended if book marks do not overlap. Because the bookmark text ends when another bookmark is detected during import, if you overlap bookmarks the results are not as you intended.
- In the DOCX file, do not cross paragraph boundaries. If the bookmark continues across two or more paragraphs, the text used for the bookmark ends at the end of the first paragraph.

- In the DOCX file, do not place bookmarks at the end of a paragraph. If the bookmark ends in the paragraph, it is kept in that paragraph on publish. The last instance of such a bookmark contains a space as its text. If the bookmark does not end in the paragraph, it is associated with the next non-empty paragraph.
- Do not use `_Toc*` in the bookmark name (in Microsoft Word or after import in Integrity Lifecycle Manager). Although the bookmarks are imported to,Integrity Lifecycle Manager, they are not included when the document is later exported to DOCX format. Instead, create a new bookmark to the figure, heading, or table that you want to reference. Editing the bookmark is also not recommended because it prevents the link in the table of contents from working. A best practice is to create the reference in the DOCX file before import. This creates `_Ref*` style bookmarks that are imported and exported without problems.
- Do not create cross-references without a hyperlink. In the DOCX file, only cross-references that use the hyperlink `(\h)` switch are converted to cross-references during import into Integrity Lifecycle Manager. A cross-reference without that switch is imported and preserved as a Microsoft Word field definition, with no clickable cross-reference link in Integrity Lifecycle Manager.
- Do not specify additional switches with cross-references if the hyperlink switch `(\h)` is specified. This is because only the hyperlink switch is retained for cross-references when publishing the source document. If additional switches are set, those do not exist in the document if exported to DOCX format.
- To ensure that shape objects containing cross-references are correctly imported in Integrity Lifecycle Manager, position such objects using **Position ▸ In Line with Text** option in Microsoft Word.
- Before import, ensure that shape objects containing cross-references and reference type elements are in different paragraphs.
- For operations related to import, edits, export, or reimport of Microsoft Word documents to Integrity Lifecycle Manager, it is recommended to use similar supported Microsoft Word versions for the same document.

Limitations For Importing a Word Document That Contains Cross-references Inside Shapes, Canvas, or Group Objects

- If a text box or a shape object contains multiple cross-references, the order of the added cross-references below the generated thumbnail image is not retained after import.
- If the shape objects contain cross-references and reference type elements in the

same paragraphs, an error for the cross-references is displayed. Before import, you must move such shape objects to the next paragraphs.

- For the documents authored in Microsoft Word 2007, Integrity Lifecycle Manager displays an error for cross-references after importing documents that contain cross-references inside shapes, canvas, or group objects. However, Integrity Lifecycle Manager creates the cross-references below the generated thumbnail image after import.

Tip

Convert the non-supported Microsoft Word versions (for example, Microsoft Word 2007) to the Microsoft Word versions supported by Integrity Lifecycle Manager and then perform the import operation.

Caption Support

The topic explains how Gateway and Integrity Lifecycle Manager support the import of captions from DOCX files. Captions include cross-references and the Table of Figures and the Table of Contents elements.

Caption Representation in Integrity

The following list describes how Gateway and Integrity Lifecycle Manager handle captions in DOC files:

- Format information of captions is preserved in Integrity Lifecycle Manager.
- Most formatting of the Table of Figures element is preserved in Integrity Lifecycle Manager. The tab leader is not retained. When exporting a document, the tab leader is replaced with a series of dots.
- When the document is exported, captions display the text that was displayed in Integrity Lifecycle Manager. The caption must be updated in an editor to display referenced information. This is particularly true if you or another use manually modifies the caption in Integrity Lifecycle Manager.
- At a minimum, a caption contains a sequence field such `{ SEQ ... }`. The document containing this field can be exported with no additional support required from the DOCX template. Captions can also contain a style reference field such as `{ STYLEREF ... }` to add a chapter number to the caption. The style reference field requires that the referenced style, typically Heading 1, exists in the DOCX template. The text with this style must also exist in the exported document. In addition, proper chapter numbering requires that the

referenced style is associated with a list level. If the referenced style is not associated with a list level, the chapter reference is always 0.

- When the document is exported, any Table of Figures element that was previously imported into Integrity Lifecycle Manager is written to the resulting DOCX file as an empty Table of Figures field. You must open the document in an editor and update the field to list all figures and correct page numbers in the exported document. As mentioned earlier, the tab leader is replaced with a series of dots.
- When the document is exported, any cross-references that were imported into Integrity Lifecycle Manager are written to the resulting DOCX file as a cross-reference field containing the text with which they were imported, such as `{ REF ... }`.
- When updating fields in an editor, updating a cross-reference to anything other than a supported reference causes the standard error for an invalid reference to display.

Key Considerations

The following is a list of key considerations when including captions in your DOCX documents:

- Do not edit captions in the Integrity Lifecycle Manager **Document** view.

  Edit captions at your own risk. Integrity Lifecycle Manager offers no guarantees on the results achieved when editing captions using any of the methods described here or elsewhere. Communicate to your Integrity Lifecycle Manager users your organizational policy regarding editing such information.

  If a caption must be modified from the Integrity Lifecycle Manager **Document** view, consider the following guidelines:

  - Type the new text inside the caption before deleting the old text.
  - When updating numbering references, type the new number after the old number and then delete the old number. For example, assume that you want to change 7-2 to 7-3. To the 7–2, add a 3 to form 7–23. Next, delete the 2 to form 7-3.
  - The numbers displayed in the Integrity Lifecycle Manager **Document** view are only text representations of the field value. Hidden field information is stored on the HTML tag containing the field value for use by Gateway. Currently, the hidden field information cannot be modified in Integrity Lifecycle Manager.
- Do not add captions in the Integrity Lifecycle Manager **Document** view.

Add captions at your own risk. Integrity Lifecycle Manager offers no guarantees on the results achieved when adding captions using any of the methods described here or elsewhere. Communicate to your Integrity Lifecycle Manager users your organizational policy regarding editing such information.

If a caption must be added in the Integrity Lifecycle Manager **Document** view, consider the following guidelines:

  - Start a new paragraph below the inserted table or figure.
  - When copying the numbering portion, ensure that the following conditions are met:
    - You are only copying from an existing caption of the appropriate type.
    - You are pasting into the appropriate location in the new paragraph that you started.

- You cannot edit a Table of Figures element or Table of Contents element from the Integrity Lifecycle Manager **Document** view.

  Integrity Lifecycle Manager uses the **Edit in Word** command to modify these elements.

  Integrity Lifecycle Manager does not support manual editing of these elements because the assumption is that the external editor automatically generates these fields.
- Do not use heading styles in captions

  Heading styles are not supported for use in captions.

List Support

This section explains how Gateway and Integrity Lifecycle Manager support the import of lists from DOCX files. PTC recommends that you identify issues with lists before importing DOCX files into Integrity Lifecycle Manager.

The examples in this section illustrate how list ordering in the imported DOCX file is generally preserved on export, even when the list ordering displays differently in Integrity Lifecycle Manager.

Key Considerations

When importing lists, consider the following:

- For an ordered list in a table in Microsoft Word to be interpreted as an ordered list in Integrity Lifecycle Manager, the list must span two or more table cells.
- Important attribute metadata that is needed for document export can be lost if you delete an entire list in Integrity Lifecycle Manager. For more information, see List Information Stored in Attributes on page 118.

- When setting the numbering value in Microsoft Word for a list, list entries created by the operation display in Integrity Lifecycle Manager, even if they are not displayed in Microsoft Word. To see the list entries in Microsoft Word, you might need to display paragraph markers. Such list entries are visible in the exported document.
- Lists created using Microsoft Word can contain invisible identities and user-defined starting values for element numbering. In other words, Word displays something differently than what is recorded underneath. During import, Gateway and Integrity Lifecycle Manager attempt to accurately represent the list after importing the document. While the import faithfully uses what is recorded underneath, some results can possibly differ from user expectations. Closely examine the examples in this section to ensure that user expectations are met. Or, if that is not possible, properly define for them.
- If the definition for a list entry is not valid in Microsoft Word, Gateway ignores the definition and treats the text as a regular paragraph. This occurs unless the invalid entry immediately follows a valid list entry. In this case, the invalid entry is considered part of the previous list.

Export of List Styles from Integrity to Microsoft Word Documents

When a list is exported to a Microsoft Word document, the list definition is dynamically generated based on the list in Integrity Lifecycle Manager. If no style is specified, a default style is applied. Lists retain the applied style when exported from Integrity Lifecycle Manager to Microsoft Word.

Note

Microsoft Word imposes a maximum of nine levels for a nested list. All numbered list entries have the label format of *n*. For example, *n* can be "1.", "b.", "C.", "iv.", or "V.". List support associated with the list instance remains the same. For example, the same label format is used for lists across table cell boundaries.

The following styles are supported:

Ordered List		**Unordered List**	
`<ol type="x" style="list-style-type:y">`		**`<ul type="x" style="list-style-type:y">`**	
x	y	x	y
1	decimal	disc	disc
a	lower-alpha	circle	circle
A	upper-alpha	square	square

Ordered List		**Unordered List**	
`<ol type="x" style="list-style-type:y">`		**`<ul type="x" style="list-style-type:y">`**	
I	lower-roman		
I	upper-roman		

By default, list definitions in Microsoft Word are constructed based on the list in Integrity Lifecycle Manager:

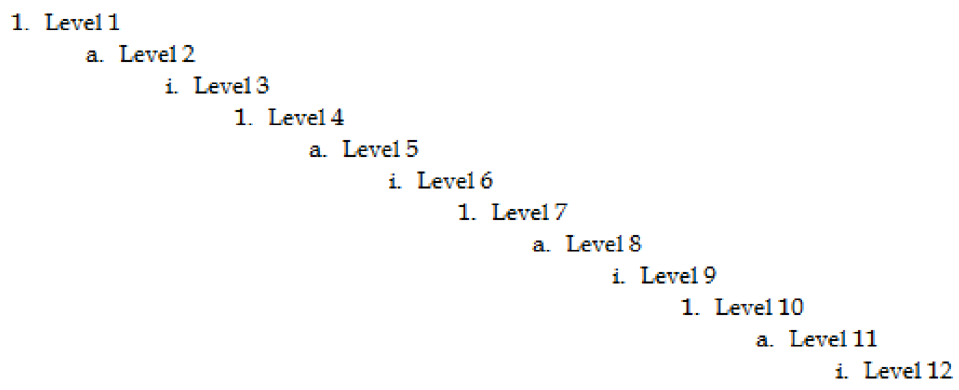
- In Integrity Lifecycle Manager, `list-style-type` can be specified explicitly through the type or style attribute. If nothing is specified, defaults are assigned as follows:
    - For ordered lists, the default is `decimal` for all levels.
    - For unordered lists, the default is based on the level:
        - Level 1: `disc`
        - Level 2: `circle`
        - Level 3: `square`
        - Subsequent levels: `square`
- If the style type is defined in both type and style, the value in style is applied.

  When importing a list from Microsoft Word to Integrity Lifecycle Manager, the style type is set in the style attribute. If you change the list property in Integrity Lifecycle Manager, the new style type is recorded with the type attribute, without updating the style attribute. This means that the list in the exported Microsoft Word document can have a different format from the list displayed in Integrity Lifecycle Manager.
- If a list has fewer than nine levels in Integrity Lifecycle Manager, default level definitions are created for the missing levels. This allows you to continue editing the list in the exported Microsoft Word document.

Note

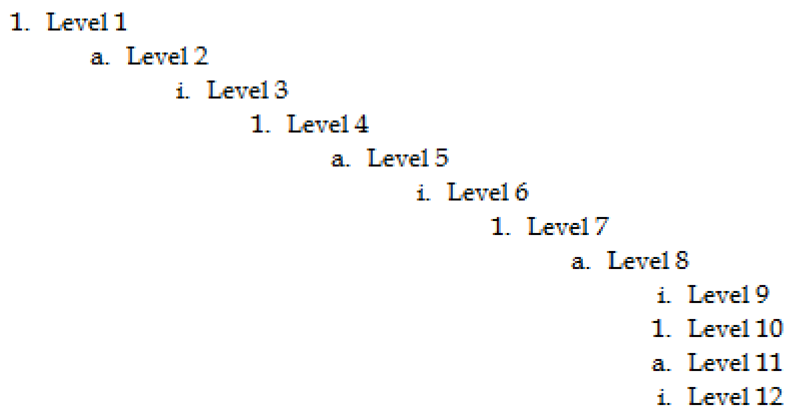
Any missing style types are set to the style type of the list's top level.

- If a list has more than nine levels in Integrity Lifecycle Manager, after export to Microsoft Word, the additional levels are treated as part of the ninth level. For example, in Integrity Lifecycle Manager, the same list displays as:



1. Level 1
    a. Level 2
        i. Level 3
            1. Level 4
                a. Level 5
                    i. Level 6
                        1. Level 7
                            a. Level 8
                                i. Level 9
                                    1. Level 10
                                        a. Level 11
                                            i. Level 12

After export to Word, the same list displays as:



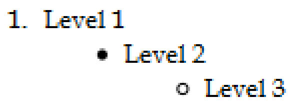
1. Level 1
    a. Level 2
        i. Level 3
            1. Level 4
                a. Level 5
                    i. Level 6
                        1. Level 7
                            a. Level 8
                                i. Level 9
                                1. Level 10
                                a. Level 11
                                i. Level 12

Known Limitations When Editing Rich Content

When creating or modifying a list in an Integrity Lifecycle Manager rich content field, the resulting HTML can display an inconsistent style when viewed in the Integrity Lifecycle Manager Web interface. The resulting HTML can also display an inconsistent style when exported to Microsoft Word. The following limitations apply when working with list styles in an Integrity Lifecycle Manager rich content field:

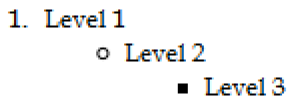
- Assume that you are working in the Integrity Lifecycle Manager client GUI with a list element that includes an invalid type attribute such as `<ol type="disc">` or `<ul type="A">`. The list entry is rendered based on the value of the type attribute. The name of the list tag is ignored. When the list is exported to Microsoft Word, the list tag takes precedence. If the value of the type is not applicable to the list element, it is ignored. This same behavior also occurs in the Integrity Lifecycle Manager Web interface.
- In the Integrity Lifecycle Manager client GUI, the style type for the top-level bulleted list is applied to a bulleted list that is a sub-list of a numbered list. When the list is exported to Microsoft Word, the level of the sub-list is considered. This same behavior also occurs in the Integrity Lifecycle Manager Web interface. Thus, bullet symbols can vary between outputs.

  For example, in the Integrity Lifecycle Manager client GUI the list displays as:



1. Level 1
    - ● Level 2
        - ○ Level 3

  After export to Microsoft Word, the same list displays as:



1. Level 1
    - ○ Level 2
        - ▪ Level 3

Sub-List Support

On export from Integrity Lifecycle Manager, sub-lists are always renumbered unless the list contains metadata derived from a previous import. Assume that you add entries to the sub-list that was previously imported. Because this sub-list contains metadata, on export, the sub-list numbering is only that which was imported originally. When metadata is present, no new numbering occurs on export.

List Information Stored in Attributes

After importing a DOCX file, Integrity Lifecycle Manager maintains list information in attributes. PTC does not intend or recommend that you modify attribute values directly. However, attribute values are visible from the command line interface when the `im viewissue --showRichContent` command option is used.

The attributes that display are `data-mksdata` and `data-mksclass`. While it is not meaningful to describe the attribute values in this context, they perform an important function and require some consideration. The attributes are used internally by Integrity Lifecycle Manager and Gateway to reconstruct the original lists correctly on export where that is stated to happen in this section's examples.

Caution

Attribute metadata can be lost if you delete the entire list. This is true even if you subsequently recreate the list manually to appear the same as the original list. If you need to edit the list, ensure that you are only deleting an entry rather than the entire list.

List metadata is included when copying and pasting list text in Integrity Lifecycle Manager. When working with imported lists in Integrity Lifecycle Manager, assume that you copy and paste the list within the same requirement that already includes a list. The pasted list is stitched into already existing list. If you paste the list into a new requirement, a second distinct list is created in the document.

Assume that there are already two lists in the same requirement. You copy the first list and paste it after the second list. The pasted list is stitched to the first list, even though the text appears after the second list. This means that the list numbering appears to skip past what was the second list and is now middle the list.

One special case to consider is when the requirement contains a table continuously numbered across table cells. Assume that you delete the list that had the imported metadata stored in attributes. When the document is exported, Gateway determines that the text is in the same style that was in the table for what was in the primary list. Gateway then adds those attributes back. The list level is lost, but the list instance exists.

Continuous List Examples

This section includes examples where Gateway determines that an ordered list is continuous across cells and therefore interprets the list as such in Integrity Lifecycle Manager. Continuous cells are useful because you can add or delete entries to them in Integrity Lifecycle Manager and the ordered list numbering is preserved.

A table is deemed to contain a continuous list if the first encountered top-level numbered list has another entry in a different cell. For examples of non-continuous lists, see Non-continuous List Examples on page 122.

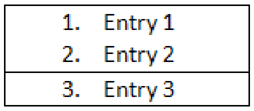
In some examples, even when an ordered list is not continuous after being imported into Integrity Lifecycle Manager, it is still continuous after exporting to DOCX format. This is because the information needed to construct the original lists is still maintained in Integrity Lifecycle Manager. For more information, see List Information Stored in Attributes on page 118.

Table cell examples are described in an `x:y` format. For example, table cell 1:2 is the first cell on the `x` axis and the second cell on the `y` axis.

Ordered List Spanning Cells

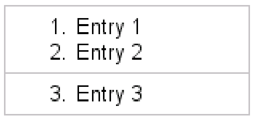
The following are examples of continuous lists:

MS Word (Original)



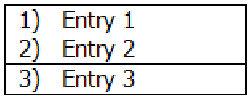
1. Entry 1 2. Entry 2
3. Entry 3

Integrity Lifecycle Manager (Imported)



1. Entry 1 2. Entry 2
3. Entry 3

MS Word (Exported)



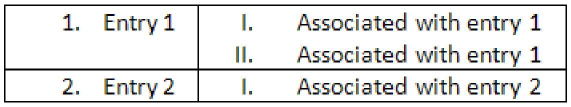
1) Entry 1 2) Entry 2
3) Entry 3

These examples are continuous lists because each ordered list spans cell 1 and cell 2. To see an example where the list is not continuous, see "Unordered List and Ordered List Example" in Non-Continuous List Examples on page 122.

Multiple Lists Spanning Cells

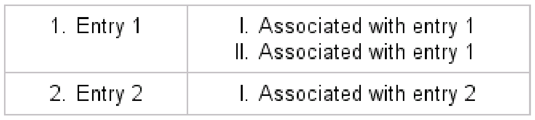
The following is a continuous list example showing sub-lists:

MS Word (Original)



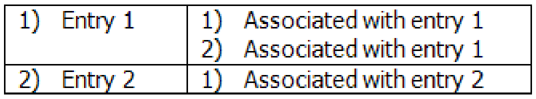
1. Entry 1	I. Associated with entry 1 II. Associated with entry 1
2. Entry 2	I. Associated with entry 2

Integrity Lifecycle Manager (Imported)



1. Entry 1	I. Associated with entry 1 II. Associated with entry 1
2. Entry 2	I. Associated with entry 2

MS Word (Exported)



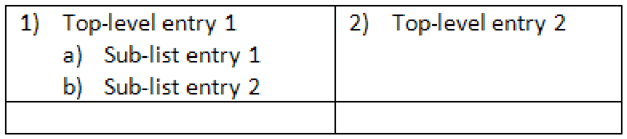
1) Entry 1	1) Associated with entry 1 2) Associated with entry 1
2) Entry 2	1) Associated with entry 2

In this example, only one of the three lists is continuous after import into Integrity Lifecycle Manager. The first list is continuous because it spans two table cells: cell 1:1 and cell 2:1. The list in cell 1:2 is associated with Entry 1 in cell 1:1. However, it is not continuous because it is not the top-level list.

Top-Level List Spanning Cells

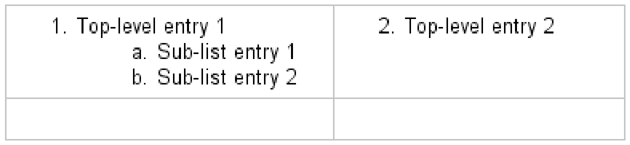
The following is an example of a list with a sub-list:

MS Word (Original)



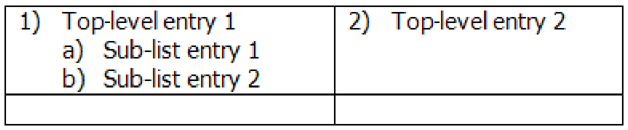
1) Top-level entry 1 a) Sub-list entry 1 b) Sub-list entry 2	2) Top-level entry 2

Integrity Lifecycle Manager (Imported)



1. Top-level entry 1 a. Sub-list entry 1 b. Sub-list entry 2	2. Top-level entry 2

MS Word (Exported)



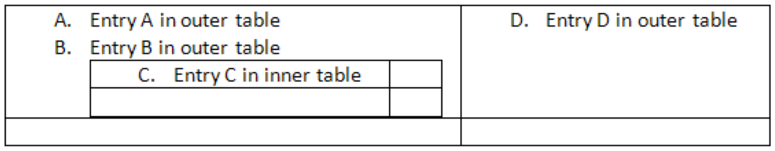
1) Top-level entry 1 a) Sub-list entry 1 b) Sub-list entry 2	2) Top-level entry 2

The top-level list is continuous because it spans cell 1:1 and cell 1:2. The list also includes a sub-list in cell 1:1. However, this list is not continuous because it is not at the top level. When adding or removing entries in Integrity Lifecycle Manager, only entries added to the continuous list are preserved.

Table Contained in a List

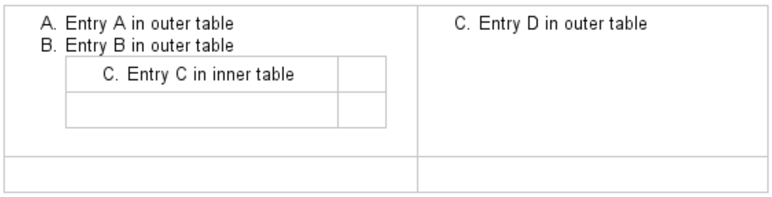
The following is an example of a list that extends into a subtable:

MS Word (Original)



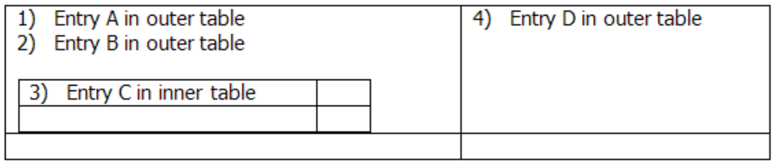
A. Entry A in outer table
B. Entry B in outer table
C. Entry C in inner table
D. Entry D in outer table

Integrity Lifecycle Manager (Imported)



A. Entry A in outer table
B. Entry B in outer table
C. Entry C in inner table
C. Entry D in outer table

MS Word (Exported)



1) Entry A in outer table
2) Entry B in outer table
3) Entry C in inner table
4) Entry D in outer table

In this example, the subtable is not continuous after importing the document into Integrity Lifecycle Manager. The list numbering ignores the subtable and continues into cell 2:1. List entries added to the subtable are numbered. However, the original numbering is reconstructed during export.

Non-Continuous List Examples

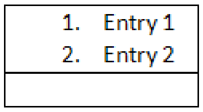
This topic contains examples where the numbering is not continuous across list entries in ways you need to be aware of before importing a document. This means that if you add entries to these lists in Integrity Lifecycle Manager, the numbering does not continue.

Table cell examples are described in an `x:y` format. For example, table cell 1:2 is the first cell on the `x` axis and the second cell on the `y` axis.

Unordered List and Ordered List Example

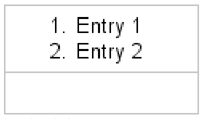
The following example starts with an unordered list extending across two cells and an ordered list contained in a single cell:

MS Word (Original)



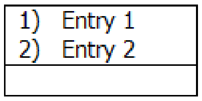
1. Entry 1 2. Entry 2

Integrity Lifecycle Manager (Imported)



1. Entry 1 2. Entry 2

MS Word (Exported)



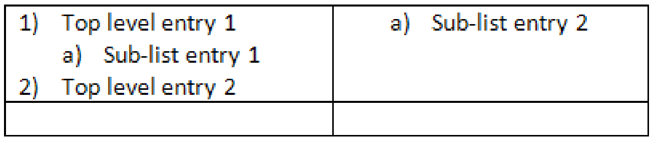
1) Entry 1 2) Entry 2

The ordered list numbering is not maintained if new entries are added because the list only exists in a single cell. Ordered lists need to extend across two or more cells in the table for Integrity Lifecycle Manager to maintain the numbering when new entries are added. For more information, see "Ordered List Spanning Cells" in Continuous List Examples on page 119. However, pre-existing ordered list entries are maintained on export.

Second Sub-List Without Parent

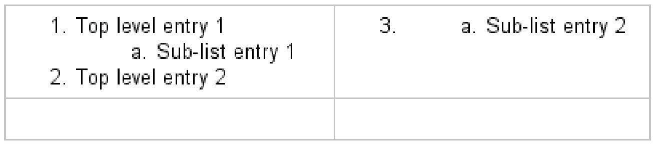
The following is an example where the top-level list does not continue to the cell 1:2, but there is a sub-list without a parent.

MS Word (Original)



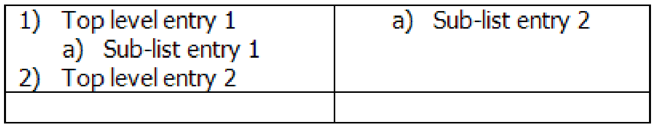
1) Top level entry 1 a) Sub-list entry 1 2) Top level entry 2	a) Sub-list entry 2

Integrity Lifecycle Manager (Imported)



1. Top level entry 1 a. Sub-list entry 1 2. Top level entry 2	3. a. Sub-list entry 2

MS Word (Exported)

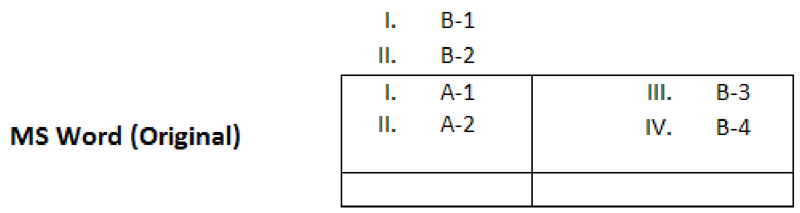


1) Top level entry 1 a) Sub-list entry 1 2) Top level entry 2	a) Sub-list entry 2

In this example, the numbering appears to be continuous when imported into Integrity Lifecycle Manager. This is because Integrity Lifecycle Manager determines that the list must continue into cell 1:2 as number 3 to preserve the numbering for Sub-list entry 2. In actuality, the list is not continuous. Consequently, numbering is not preserved when entries are added or removed in Integrity Lifecycle Manager. However, the original numbering is reconstructed when the document is exported.

List Starts Outside Table

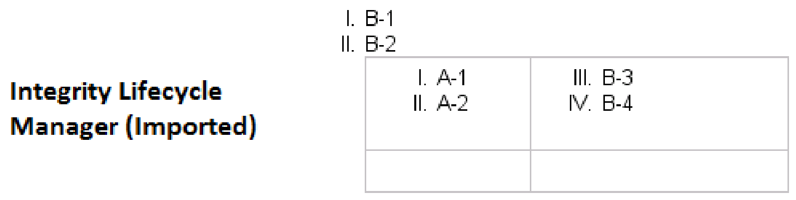
The following is an example where the list in cell 1:2 continues the list started outside of the table (B-1 to B-4). You can see how it differs from when the list starts in cell 1:1 (A-1 to A-2).



MS Word (Original)

I. B-1
II. B-2

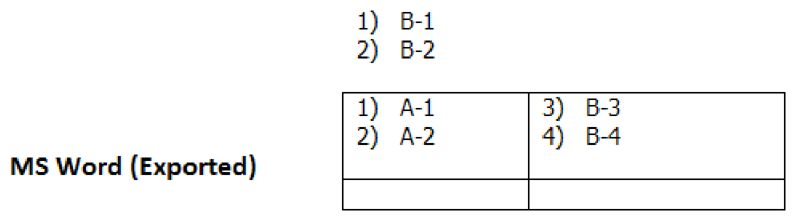
I. A-1 II. A-2	III. B-3 IV. B-4



Integrity Lifecycle Manager (Imported)

I. B-1
II. B-2

I. A-1 II. A-2	III. B-3 IV. B-4



MS Word (Exported)

1) B-1
2) B-2

1) A-1 2) A-2	3) B-3 4) B-4

In this example, the table appears to contain a single ordered list. In actuality, there are two lists in the table, and neither list is continuous. Even though the list in cell 1:2 continued from a list starting outside the table, the list is not continuous. This is because the list must span two or more cells to be continuous. However, on export, the correct numbering is reconstructed.

List Starting Number Examples

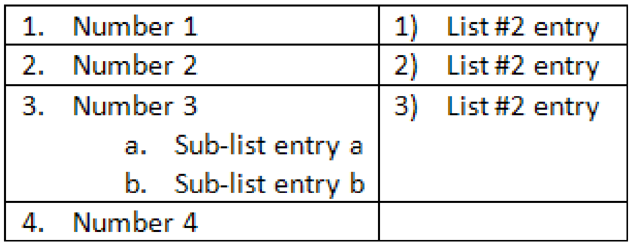
This topic contains examples that illustrate list numbering for Microsoft Word documents that are imported and exported using Gateway.

Table cell examples are described in an x:y format. For example, table cell 1:2 is the first cell on the x axis and the second cell on the y axis.

Primary List Example with Spanning

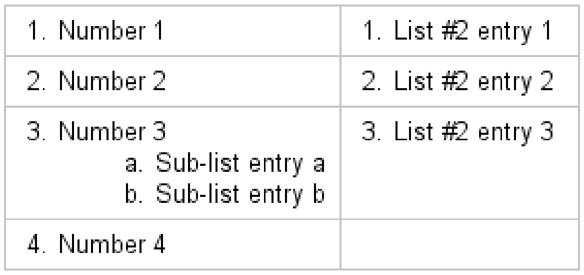
The following is an example where the primary list spans multiple cells. A sub-list is confined to a single cell A second list spans multiple cells.

MS Word (Original)



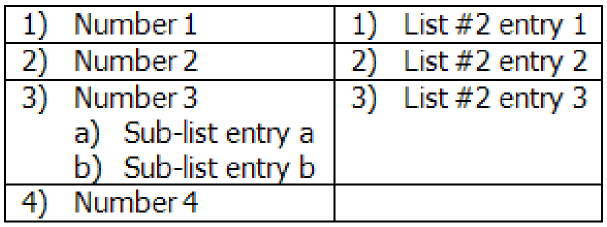
1. Number 1	1) List #2 entry
2. Number 2	2) List #2 entry
3. Number 3 a. Sub-list entry a b. Sub-list entry b	3) List #2 entry
4. Number 4	

Integrity Lifecycle Manager (Imported)



1. Number 1	1. List #2 entry 1
2. Number 2	2. List #2 entry 2
3. Number 3 a. Sub-list entry a b. Sub-list entry b	3. List #2 entry 3
4. Number 4	

MS Word (Exported)



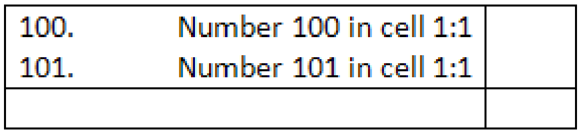
1) Number 1	1) List #2 entry 1
2) Number 2	2) List #2 entry 2
3) Number 3 a) Sub-list entry a b) Sub-list entry b	3) List #2 entry 3
4) Number 4	

The list in the first column is the primary list. Its numbering is retained if you add or delete entries. The list in the second column is not the primary list. Consequently, its numbering is not retained if you add or delete entries.

Arbitrary Start Number Example

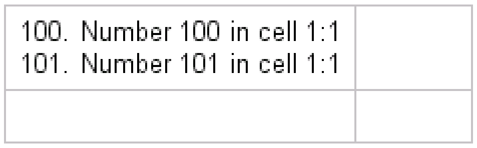
The following is an example of how a list start number is preserved within a table cell:

MS Word (Original)



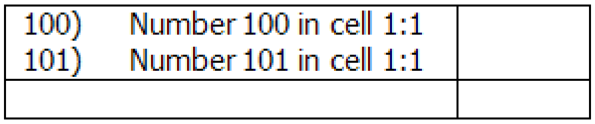
100. Number 100 in cell 1:1 101. Number 101 in cell 1:1	

Integrity Lifecycle Manager (Imported)



100. Number 100 in cell 1:1 101. Number 101 in cell 1:1	

MS Word (Exported)

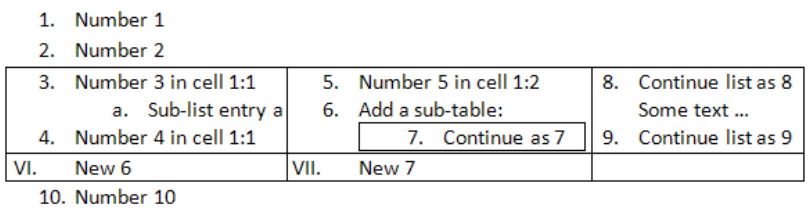


100) Number 100 in cell 1:1 101) Number 101 in cell 1:1	

Table with a Nested Table

The following is an example of numbering starting outside a table and then across a nested table:

MS Word (Original)

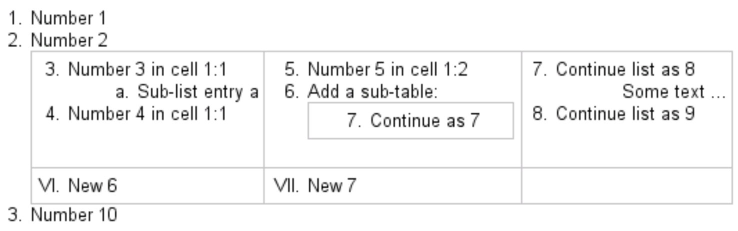


1. Number 1
2. Number 2

3. Number 3 in cell 1:1 a. Sub-list entry a 4. Number 4 in cell 1:1	5. Number 5 in cell 1:2 6. Add a sub-table: 7. Continue as 7	8. Continue list as 8 Some text ... 9. Continue list as 9
VI. New 6	VII. New 7	

10. Number 10

Integrity Lifecycle Manager (Imported)

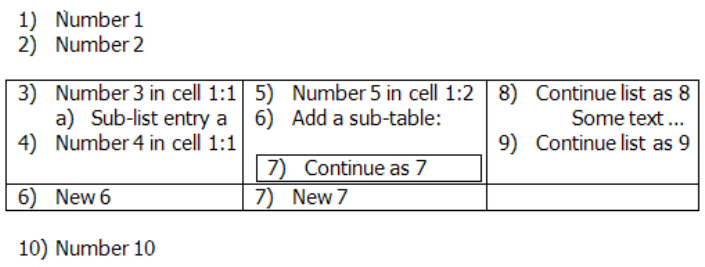


1. Number 1
2. Number 2

3. Number 3 in cell 1:1 a. Sub-list entry a 4. Number 4 in cell 1:1	5. Number 5 in cell 1:2 6. Add a sub-table: 7. Continue as 7	7. Continue list as 8 Some text ... 8. Continue list as 9
VI. New 6	VII. New 7	

3. Number 10

MS Word (Exported)



1) Number 1
2) Number 2

3) Number 3 in cell 1:1 a) Sub-list entry a 4) Number 4 in cell 1:1	5) Number 5 in cell 1:2 6) Add a sub-table: 7) Continue as 7	8) Continue list as 8 Some text ... 9) Continue list as 9
6) New 6	7) New 7	

10) Number 10

The table is considered part of the list started before the table. After the document is imported in Integrity Lifecycle Manager, the list entries outside of the table are considered the continuation of the list containing the table. Consequently, 10 becomes 3.

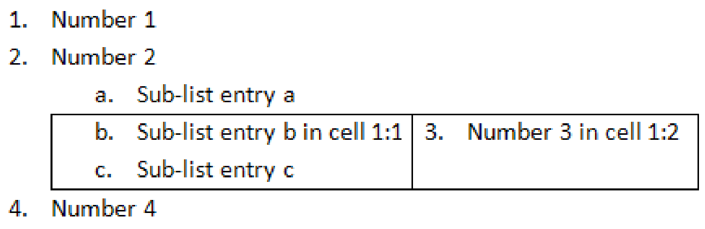
The numbering within the table continues, ignoring the subtable. Consequently, 8 becomes 7, and 9 becomes 8.

When exported, the original list numbering is reconstructed correctly.

Table List Starting Outside the Table with a Missing Entry

The following is an example of a list starting outside of the table but continuing into the second cell of the table:

MS Word (Original)

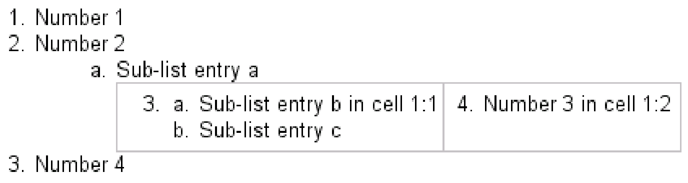


1. Number 1
2. Number 2
   a. Sub-list entry a

b. Sub-list entry b in cell 1:1 c. Sub-list entry c	3. Number 3 in cell 1:2

4. Number 4

Integrity Lifecycle Manager (Imported)

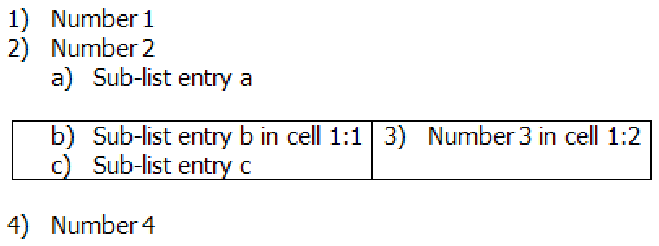


1. Number 1
2. Number 2
   a. Sub-list entry a

3. a. Sub-list entry b in cell 1:1 b. Sub-list entry c	4. Number 3 in cell 1:2

3. Number 4

MS Word (Exported)



1) Number 1
2) Number 2
   a) Sub-list entry a

b) Sub-list entry b in cell 1:1 c) Sub-list entry c	3) Number 3 in cell 1:2

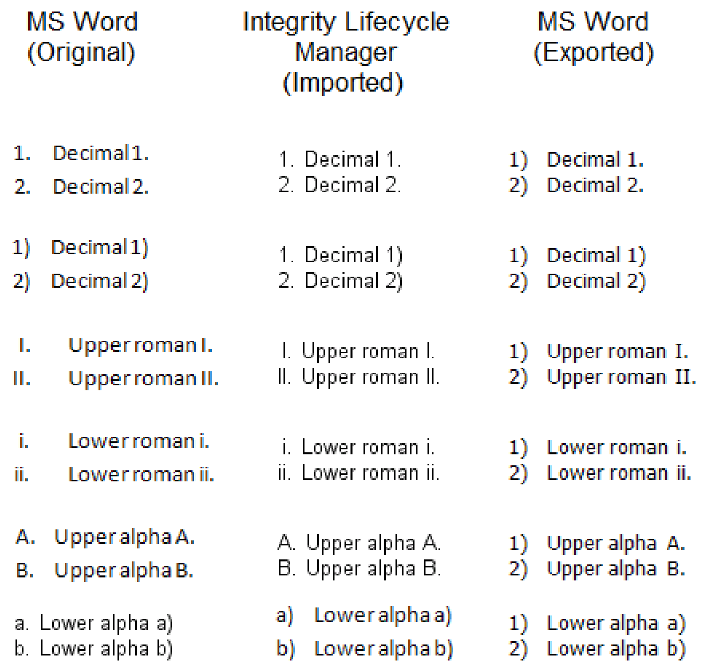
4) Number 4

The table contains a list that was not started at the top level. When imported into Integrity Lifecycle Manager, the sub-list is considered to have its parent level missing. Consequently, 3 is inserted into cell 1:1, and in cell 1:2, the number 3 becomes 4. The list in table cell 1:1 and cell 1:2 appears to start at 3 and be continuous. In actuality, it is not continuous because Integrity Lifecycle Manager added the 3. The 3 as not already present.

When the document is exported, the original list numbering is reconstructed correctly.

List Type Support

The following example demonstrates several list instances that illustrate support for list types in DOCX files:

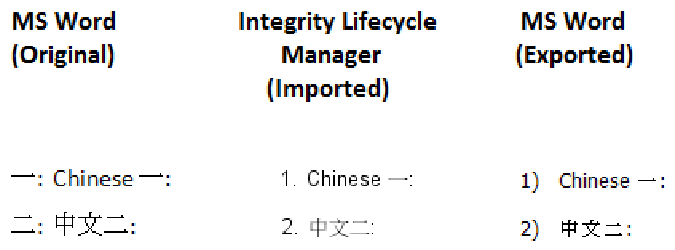


MS Word (Original)	Integrity Lifecycle Manager (Imported)	MS Word (Exported)
1. Decimal 1. 2. Decimal 2.	1. Decimal 1. 2. Decimal 2.	1) Decimal 1. 2) Decimal 2.
1) Decimal 1) 2) Decimal 2)	1. Decimal 1) 2. Decimal 2)	1) Decimal 1) 2) Decimal 2)
I. Upper roman I. II. Upper roman II.	I. Upper roman I. II. Upper roman II.	1) Upper roman I. 2) Upper roman II.
i. Lower roman i. ii. Lower roman ii.	i. Lower roman i. ii. Lower roman ii.	1) Lower roman i. 2) Lower roman ii.
A. Upper alpha A. B. Upper alpha B.	A. Upper alpha A. B. Upper alpha B.	1) Upper alpha A. 2) Upper alpha B.
a. Lower alpha a) b. Lower alpha b)	a) Lower alpha a) b) Lower alpha b)	1) Lower alpha a) 2) Lower alpha b)

During import, Gateway attempts to match the list type to one of the following: decimal, upper roman, lower roman, upper alpha, and lower alpha. If there is a type that cannot be matched to one of these types, Gateway converts it to the decimal type on import.

Numbering types are not maintained during export.

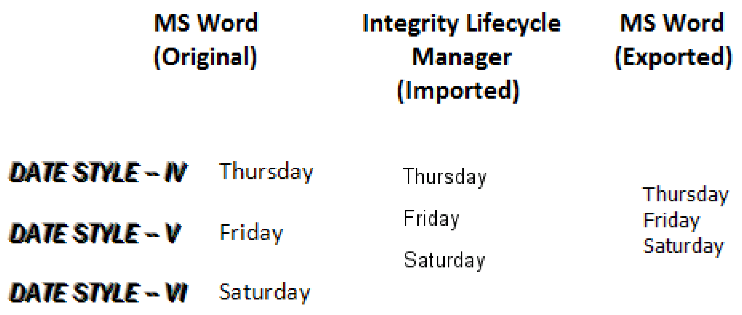
The following is an example where the type is converted to decimal during import:



MS Word (Original)	Integrity Lifecycle Manager (Imported)	MS Word (Exported)
一: Chinese 一: 二: 中文二:	1. Chinese 一: 2. 中文二:	1) Chinese 一: 2) 中文二:

Lists that Link a Level to a Style

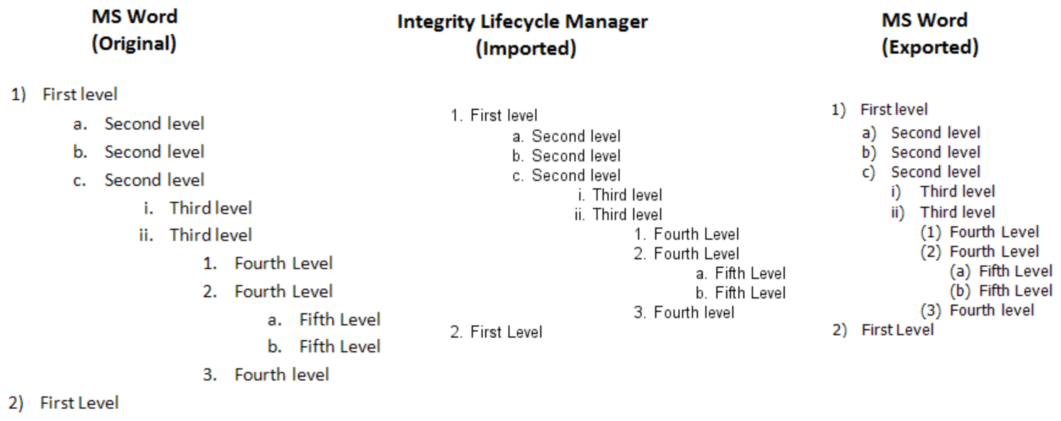
Lists that link a level to a style are not supported. The following example demonstrates such a list:



	MS Word (Original)	Integrity Lifecycle Manager (Imported)	MS Word (Exported)
DATE STYLE – IV	Thursday	Thursday	Thursday
DATE STYLE – V	Friday	Friday	Friday
DATE STYLE – VI	Saturday	Saturday	Saturday

Multiple-Level Lists

The following example illustrates how lists with multiple levels are retained or converted during import and export:



MS Word (Original)

```
1)  First level
        a.  Second level
        b.  Second level
        c.  Second level
                i.  Third level
               ii.  Third level
                        1.  Fourth Level
                        2.  Fourth Level
                                a.  Fifth Level
                                b.  Fifth Level
                        3.  Fourth level
2)  First Level
```

Integrity Lifecycle Manager (Imported)

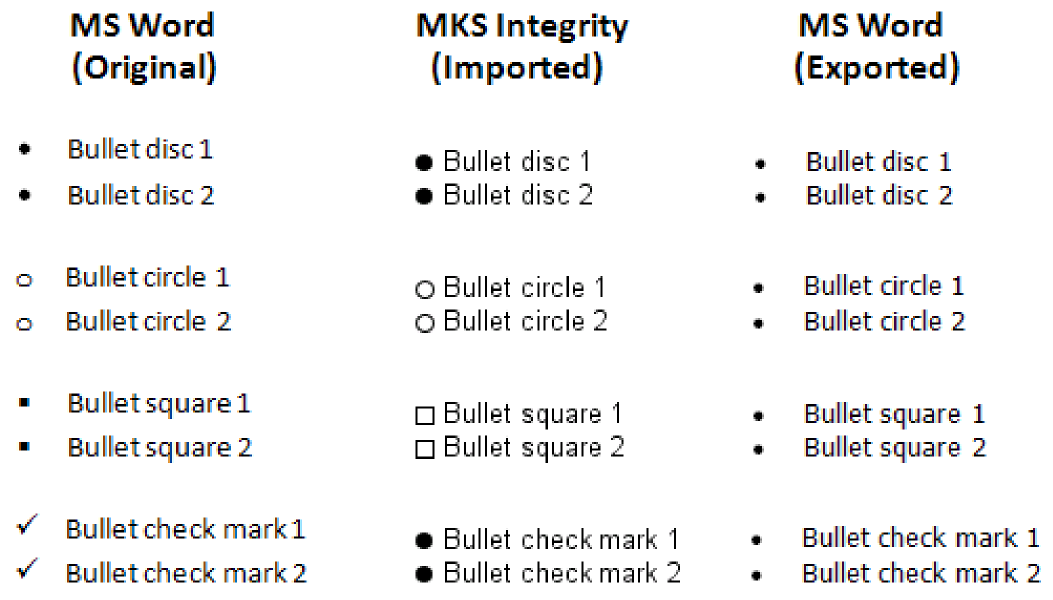
```
1. First level
      a. Second level
      b. Second level
      c. Second level
            i. Third level
           ii. Third level
                  1. Fourth Level
                  2. Fourth Level
                        a. Fifth Level
                        b. Fifth Level
                  3. Fourth level
2. First Level
```

MS Word (Exported)

```
1)  First level
    a)  Second level
    b)  Second level
    c)  Second level
        i)   Third level
        ii)  Third level
            (1)  Fourth Level
            (2)  Fourth Level
                (a)  Fifth Level
                (b)  Fifth Level
            (3)  Fourth level
2)  First Level
```

Support of Bulleted Lists

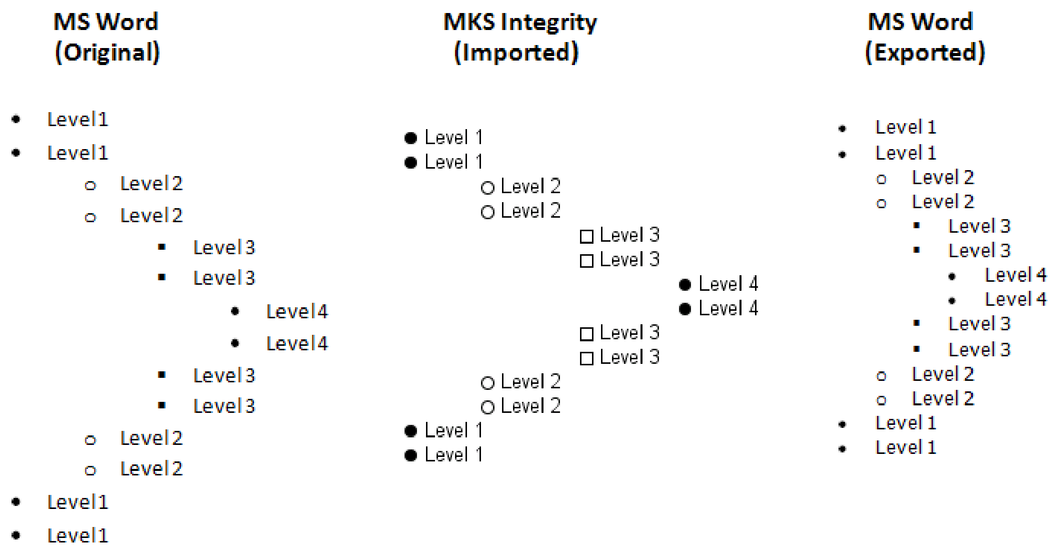
The following example contains four lists that demonstrate the support for bulleted lists:



MS Word (Original)	MKS Integrity (Imported)	MS Word (Exported)
• Bullet disc 1	● Bullet disc 1	• Bullet disc 1
• Bullet disc 2	● Bullet disc 2	• Bullet disc 2
o Bullet circle 1	○ Bullet circle 1	• Bullet circle 1
o Bullet circle 2	○ Bullet circle 2	• Bullet circle 2
▪ Bullet square 1	□ Bullet square 1	• Bullet square 1
▪ Bullet square 2	□ Bullet square 2	• Bullet square 2
✓ Bullet check mark 1	● Bullet check mark 1	• Bullet check mark 1
✓ Bullet check mark 2	● Bullet check mark 2	• Bullet check mark 2

The disc, circle, and square bullets are retained during import. Other bullets, such as the check mark, are not maintained.

The following example demonstrates the support for bulleted lists with multiple levels:



MS Word (Original)	MKS Integrity (Imported)	MS Word (Exported)
• Level 1	● Level 1	• Level 1
• Level 1	● Level 1	• Level 1
o Level 2	○ Level 2	o Level 2
o Level 2	○ Level 2	o Level 2
▪ Level 3	□ Level 3	▪ Level 3
▪ Level 3	□ Level 3	▪ Level 3
• Level 4	● Level 4	• Level 4
• Level 4	● Level 4	• Level 4
▪ Level 3	□ Level 3	▪ Level 3
▪ Level 3	□ Level 3	▪ Level 3
o Level 2	○ Level 2	o Level 2
o Level 2	○ Level 2	o Level 2
• Level 1	● Level 1	• Level 1
• Level 1	● Level 1	• Level 1

The example shows how the bullet characters used for each level on export appear based on the level. This occurs even if the bullets in the Microsoft Word document are different.