Search documentation Search

Appian

Application Deployment Guidelines

Appian 7.7

Toggle Us

Appian allows you to move process applications to different servers by exporting objects into XML files, bundling them into a ZIP-format package, and then importing them. The tool exports the associated object properties, security role maps, and information about related objects as part of the package. The links between related objects are preserved across the move except where specifically noted. In most cases, the related objects themselves are not automatically exported. (They must be explicitly added as objects within the application, or be already present on target server.)

The Application Builder allows you to scan your application for missing dependencies and automatically add them to the package. The export limitatio described in this help topic page explain those relationships that cannot be preserved during export and import of the application.

Each such limitation also applies to dependency scans. If we do not preserve an object relationship during export, it cannot be detected by the dependency scan.

See also: Application Builder and Missing Dependencies

Tu

The Export and Import Process

The export and import process follows rules for each object exported, and certain general rules.

1. Cautions
2. General Rules
   - Updates
   - Issues During Export or Import
   - Reading the Import Log
   - Object Creator and Creation/Last Modified Timestamp
   - System Configurations
   - References to Users
3. Object-Specific Rules
   - Applications
   - Discussions
   - Documents
   - Pages
   - Channels
   - Groups
   - Processes
   - Data Stores
   - Reports
   - Rules
   - Feeds
   - System Objects
   - Notes for Application Administrators
4. Security
   - Assigning Administrator Rights to the User Who Imports an Application
   - Deploying User Rights with Exported Objects
   - User Rights Deployment Example

Re

Ot

Cautions

- Modifying the generated XML files and internal structure of the ZIP export package is not supported.
- Application packages from an earlier version can be imported by later versions. Packages exported from later versions of Appian are not compatil for importing into prior versions. If your version contains a hotfix, it is considered a later version (for example, 6.7 Hotfix 2 is later than 6.7 with n hotfix installed).
- Dependencies of one application can impact another application. Importing a sub-process, group, folder, or other application component used by more than one application can have unintended side effects.
- When importing pre-7.4 applications to a 7.4+ environment, the individual actions associated with those applications will have unique URLs that can be bookmarked. If the same pre-7.4 applications are imported multiple times into a 7.4+ environment, the URLs for the associated actions w change every time the applications are imported again, so any previously bookmarked URLs will no longer work. This will happen any time a pre-7.4 application that is already on a 7.4+ environment is imported again. To avoid this inconvenience, be sure to import all pre-7.4 applications before making the environment available to users and do not re-import the same applications so that users can bookmark their actions and thos bookmarks will continue to work.

General Rules

- During export, Appian processes the list of objects to be exported one-by-one and captures the state of the object when the export is initiated. the contents of a container object change while you are creating the package, the exported contents will reflect the objects at the time the expo was initiated for that object; subsequent changes will not be included.
- It is possible to import the same package multiple times.

- It is not possible to undo changes from an import.
- Objects can be exported from and imported onto the same server for backup and restoration.

Updates

If an imported object already exists on the target system (identified by a universally unique ID or "UUID") a new version is created – so long as versioni is supported for the object. (Versioning is supported by Appian for objects such as rules, process models and documents.) When this is true, a new version is created even if only the role map of the object has changed. There is no distinction between an object's role map and its properties during import.

When versioning is not supported for an object, the imported object overwrites the object on the target system and its properties.

Issues During Export or Import

Best Practice: Test each application package using the Inspect Package feature prior to deploying the application to identify issues with a deployment before you apply changes that cannot be undone or may cause problems on the server.

- See also: Inspect Package

Export or Import issues might be encountered for various reasons.

- For example, an error may occur when you do not have ownership of the exported or imported object, or if you attempt to import a group witl the same name as another group.
- In each case an error occurs, Appian gathers information on any problems and displays it to you as a list after all objects in the package have bee processed.
- The export and import processes do not stop when errors are encountered on specific objects.

A problem during the import of an object might cause the import of related objects to also fail.

- For example, a failure to create a group type during an import would cause the import of any groups of that type to also fail (the tool does not attempt to import these groups in this case).

When an import/export problem occurs, the message displayed indicates the cause of the problem (in most cases) and identifies the object or objects that were not successfully exported or imported. In such cases, you can do one of the following:

1. Resolve the cause of the problem and export or import the package again.
2. Export or import only those objects that failed due to the issue identified. You might prefer this option when the export/import package is especially large, such that exporting or importing a subset of objects is much faster.
3. (Import only) Manually create or update the objects that could not be imported in the new environment.

If a group fails to be created during an import, and the group exists in role maps for other objects - or is a member of another group - we recommend re-importing the entire package unless it is possible for you to identify and re-import only those objects that are affected by the failed creation of the group. Any groups that don't exist on the target system when an import is completed are dropped from role maps or groups they belonged to.

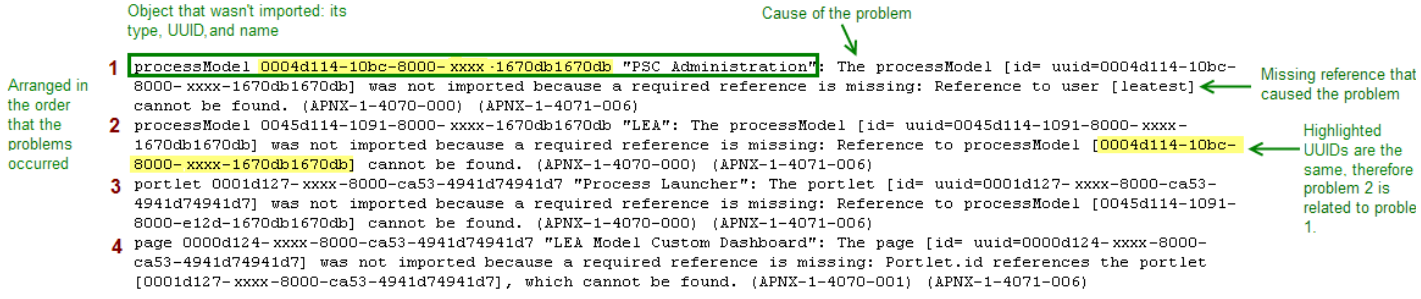
See below: Deploying User Rights with Exported Objects

Reading the Import Log

Log entries are placed in one of two sections:

- **Primary Problems**: The most important to review as it lists the items that could not be imported and the reason why. Make sure to address the cause of the problem and attempt to import the application again.
- **Cascading Problems**: Lists the items that could not be imported because they reference an item that failed to import previously. They do not need to be addressed individually.

The following example illustrates the log entries that may be generated when an object reference is not included in an import package, and is not preser on the target system.



In this instance, an imported process model listed an individual user for the assignment of an activity. We recommend using groups for assignments instead. Otherwise, ensure that the target system has a user with the same username, prior to importing the application.

Object Creator and Creation/Last Modified Timestamp

When an object is created during an import, the user performing the import is listed as its creator. The current date and time is the object's creation timestamp. When an object is updated during an import, the user performing the import is listed in the object's `last modified by` field. The current date and time is the object's `last modified` timestamp.

If the imported object supports versioning, the user performing the import is listed as the new version's creator. The current date and time is listed as th

new version's creation timestamp. The creator and creation timestamp of previous versions are not affected.

The creator and last modified by user are not given administrator rights for that object. The list of user rights are based on the security role map define for the importing object.

See below: Deploying User Rights with Exported Objects

System Configurations

System configurations cannot be exported and imported. Examples of system configurations that cannot be exported or imported include:

- Internationalization settings
- Process calendar
- Remote portlets configurations
- Global home page
- Primary navigation
- Exposing a process model as a web-service
- Rules creation security
- Rules root folder security
  ###
  References to Users ###

Users cannot be exported and imported. When importing objects that reference users such as in task assignment or constant values, the users must already be present on the target system (identified by username) for the import to succeed. Otherwise, the import of the object fails.

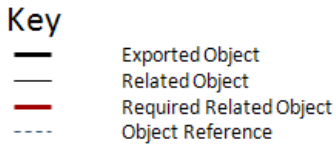
The exception is when users are referenced in object role maps or as group members.

Object-Specific Rules

Appian supports export/import for the following objects.

- Applications
- Discussions
- Documents
- Pages
- People
- Processes
- Reports
- Rules

For the images displayed in the subsections below, the following key applies:

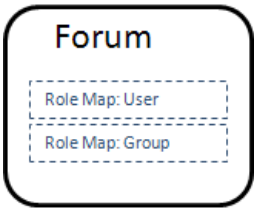


Applications

In Appian, you can export an application for import onto another server. The application properties, navigation items, user rights, and all associated objects are included in the export. For example, a process model and a page are associated with an application named **expenses**. When exporting the **expenses** application, the process model and the page are also exported. Importing an application overrides its application properties, its set of associated objects, and the existing navigation items.

Both published and unpublished applications can be exported. On import, the state of an application is the same as it was when exported.

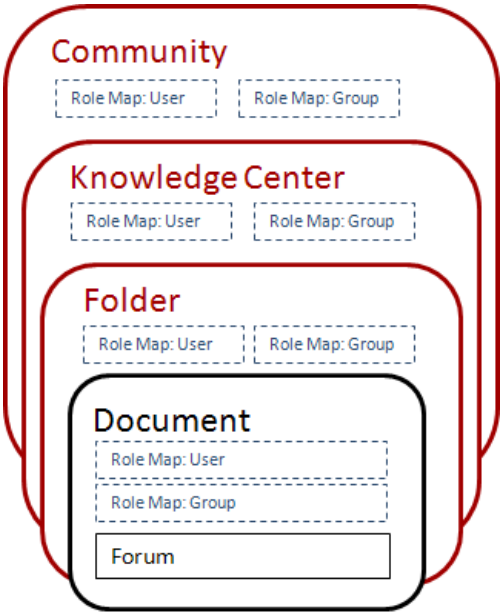
Discussions



When exporting discussion forums for import onto another server, the forum properties and user rights are included in the export.

- User subscriptions to forums are not changed if a forum is updated.
- Individual discussion topics and messages cannot be exported; just the forum container is exported.
- When a forum is updated on import, its discussion topics and messages are not updated or changed.

Documents



In Appian, a document (or file) resides in a folder, which resides in a Knowledge Center, which resides in a Community, which might reside within another Community. A document export includes the document, the document properties, and a reference to its parent object (such as the folder where the document resides). A Document, Folder, Knowledge Center, or Community might also be associated with a forum.

The image above illustrates the relationships between Document Management objects. Items displayed in red indicate required objects. Dashed lines indicate object references. Objects and references fully contained within an object's border indicate items that are automatically included within an object when it is exported.

The objects in the hierarchy must each be selected for inclusion in your application. An export of a document does not include the actual parent object of the document, only the awareness of the parent object. Imports fail if the necessary parent object is not present.

If a document already exists on the server that you are importing onto, a new version of the document is created. For documents and document containers, any existing object properties are overwritten by an import.

When exporting a content container (such as a Community, Knowledge Center, or Folder) only the selected container is exported. The contents are not exported.

If you do not have the necessary user rights to add or update the object being imported, then the import fails. For example, if an import package contains a folder that resides under a given Knowledge Center, the user importing the folder must have the right to create a folder in that Knowledge Center.

The ZIP file of Knowledge Center or Folder downloaded from the Document Management interface using the **Download** toolbar action is not importable from the Applications view. Instead, perform a bulk upload from the Documents tab.

See also: Document Management

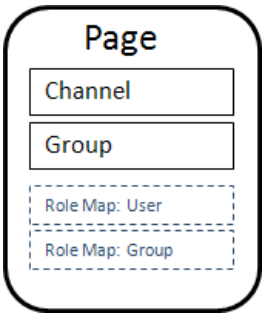
If you've created Document Management objects using the Appian API, be aware that the `system` flag is no longer used. It is ignored when importing documents and document containers.

Pages

You can export a page as part of an application, which can then be imported onto another server.

Page Export Contents

The page properties, user rights, channels, and information regarding related objects are included in the export.



Draft Pages are Exported

Pages exist in one of three states: Published, Published with Draft, and Draft (never published). When exporting, the published version of the page is exported (if available). Otherwise, the page draft is exported without its channels (the page properties and user rights are exported).

Channels are Exported

All channels (which exist on a published page) are exported with a page. Only certain channels retain configuration settings. Channels on draft pages are not exported.

Categories are Not Exported

If a page belongs to a page category, the page category field is not exported. When importing the page, if the page is created on the server for the firs time, the page does not have a page category. If the page already exists on the server, and the existing page belongs to a page category, the page category is preserved during the import.

Associated Groups can be Exported with a Page

If a page is associated with a group, the group's administrators are administrators of the page upon import. The group members are added as viewers o the page. This is the same behavior as when a team or department page is created from the user interface. Individual users cannot be exported with a page.

Page Import Rules

When importing a page onto a server that already contains the same page (identified by UUID), the properties and channels of the existing page are overwritten by the import package. If the page does not exist on the server, a new page is created with the information from the import package. Upo import, the page is always published, regardless of the state of the page that was exported.

During an import, if a corresponding page exists on the server but the page is locked, the import of the page fails. The page must be unlocked before attempting to import the page again.

When importing a custom page (a page with contents defined by a URL) the URL is not validated. The page's URL must be present for a successful import.

Channels

Channels are exported together with the page that contains them.

When importing a page, the channels that exist on the page are imported as well. If the channel also supports export and import, it is configured when imported (with some restrictions). When an imported page contains channels that do not support export/import, those channels appear on the page ir an unconfigured state.

When an imported page already exists on the target server (identified by a universally unique identifier, or UUID) the existing channels are replaced by the channels in the import package. If the replaced channel was not shared, it gets deleted. A replaced channel that has been shared for use on other pages is deactivated rather than deleted.

Channel References

Some channels reference other objects in Appian, such as process models and report documents. Any referenced objects must be independently selected for export and import. The referenced objects are not automatically included in the export package. When importing channels that rely on other objects, those objects (identified by UUID) must already exist on the target server, or be present in the import package.

Channels that reference objects that cannot be exported and imported (such as process instances or discussion topics) are imported without their prio configuration settings.

Channel Sharing

An exported shared channel remains shared when it is imported. When several pages use the same shared channel, any update included in the import package is displayed on all pages. If you do not have sufficient user rights to update the channel upon import, but you have sufficient user rights to update the page, the channel fails to import but the page imports successfully.

When a channel's sharing status changes from shared to unshared, a new UUID is given to that channel. Importing over the old version of that channe no longer changes that channel. The role map for the channel is taken from the import package, and is not merged.

If you want to remove an "unshared" channel (that has been imported) from all pages where it appears, you must reimport all the pages.

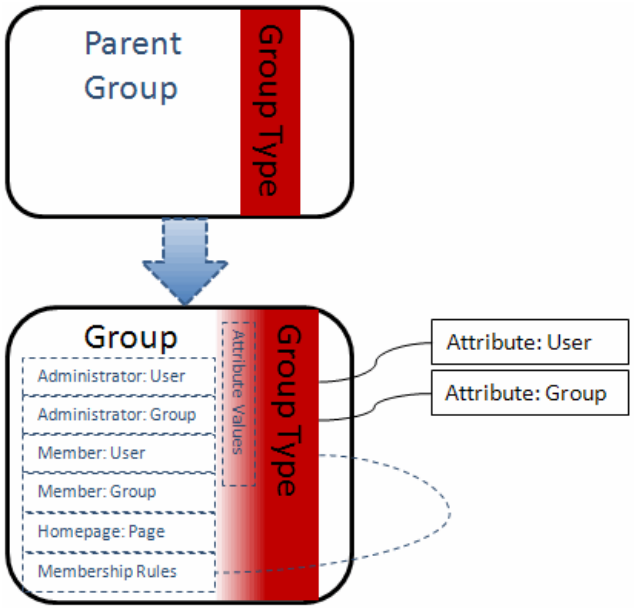
Channels Enabled for Export and Import

The following channel types support export and import, with some restrictions. All other channels are recreated in an unconfigured state when importec on a page.

Supported Channels	Restrictions
Process Launcher	If the user executing the import does not have at least Viewer security rights for the process model and the portal page (if applicable) referenced by the channel, the channel is still configured correctly on import. When the importer navigates to the page containing the channel, the channel shows a message stating that the user does not have sufficient rights to view the contents of the channel. This behavior is the same as with a process launcher channel that was not imported.
Quick Task	If the channel is configured to point to a process instance (rather than the dashboard context option) the channel configuration is not preserved through the export/import. This happens because the export/import of a process instance is not supported. If a channel references specific quick tasks (based on a selected process instance) the list of selected quick tasks is not preserved through the export/import.
Report Channel	If the user importing a report channel does not have at least Read security rights on the report document referenced by the Report channel, the channel is still configured correctly on import. When the importer navigates to the page containing the channel, the channel displays a message stating that the user does not have sufficient rights to see the report. This behavior is the same as with a report channel that was not imported.

Upload Document	
Web Content, Web Content with Process Details	• Only the published version of the channel content is exported. Version history is not exported. On import, the imported content of the channel is published. If a draft exists on the target system and the content is locked for editing, the lock is broken and the draft is lost (upon import). • Links to Appian objects in the body of Web Content channels, as well as images inserted from Knowledge Management, no longer function after the channel is imported to another installation. This limitation applies even if the referenced objects are included in the same export package. The links and images must be reconfigured to link to the correct objects on the target system. • You must remember to add any Rules or Constants used by this channel. Any Rules and Constants used do not appear on the list of missing dependencies returned when scanning your application. On import, the Rules and Constants referenced in the Web Page channel must be present in the target system or the import package.
Web Page	
My Subscribed Topics	
My Tasks	
What's New in Your Subscriptions	

Groups



Group objects (groups and group types) can be exported and imported onto another server. The object properties and information regarding related objects are included in the export, such as a user's supervisor, and a group's parent. On import, existing object properties are overwritten by the impo package. The group object's export relationships are shown in the following diagram.

In this image, solid borders on related objects (or attributes) indicate that the object can be exported together with the group. Broken lines on the related objects indicate that only a reference is exported. The related item must also exist on the target system.

When parent groups are added to an application and exported with a child group, the parent-child relationship is retained.

Groups

An exported group includes the list of all its members and administrators (users and groups), as well as information regarding the group type.

- A group's type must exist on the target server (or in the import package) for a successful import of that group. (Examples of group types includ departments and teams.)
- Group membership rules are included in export packages.
- If an imported group and an existing group are the same (based on UUID) yet have different members, the lists are combined.
  - For example, GroupA contains UserA and UserB in the import package, but on the target server, GroupA only contains UserA. Once the import takes place, UserB is added to GroupA. If UserB does not exist on the target server, is not present in the import package, or has been deactivated, the user is not added.
  - The same applies for member groups (matched by UUID) of the exported group.
- Group membership is not removed as part of application deployment.
  - Importing a group with fewer members than on the target environment does not remove the extra members on the target environment.
- If during a group import, you do not have the right to update the properties of the group, or add a user to a group, the import fails (for that group only).

- If an imported group has a parent that would violate the parent-child group relationship, the parent field is cleared. A child group cannot be set as the parent of its parent group.
- Public groups can only be created by system administrators. Non-system administrators can only create Public groups within groups that they administrate (when the `Delegated Creation` option is selected). This import behavior is the same as the behavior of the user interface.
- If a group with an existing name is being imported and duplicate names are not allowed, the import of the group fails.
  - Your server manager can specify whether group names must be unique by setting the `server.conf.personalization.personal_group_names_unique` custom property.
- If a group has a dependency on an existing group homepage, that homepage must be present in your application package for the group to be imported.

See also: Configuring User Management Default Settings

Group Types

Group types allow you to specify attributes (required or optional) for each group of that type. Required attributes must be input when each new group (of the type) is created. It is possible for these attributes to vary from system to system for the same group type. If the list of attributes of an imported group does not match the list of attributes of the group type on the target system - *and* the import package does not include a corresponding group type - the imported group's list of attributes is updated to match the group type on the target system. All groups of a group type must have the same set of attributes.

Exporting a group type packages its properties (the list of attributes and attribute values associated with all groups in the group type) into a zip file for import on another system.

When an import package includes a group and its group type - *but* the list of attributes of the group do not match the target system - the import package attributes are applied to the target system. The group type is imported first, followed by the group.

If you exclude the group type from the export package, and an imported group's attributes do not match those of the existing group, the result of the import preserves the values of the attributes on the target system.

On import, if the same group type exists on the target server (identified by UUID), all group type properties and its list of attributes are overwritten by the imported group type, except the values of the attributes. Attribute values are not overwritten on import. You cannot update the default values of group type attributes.

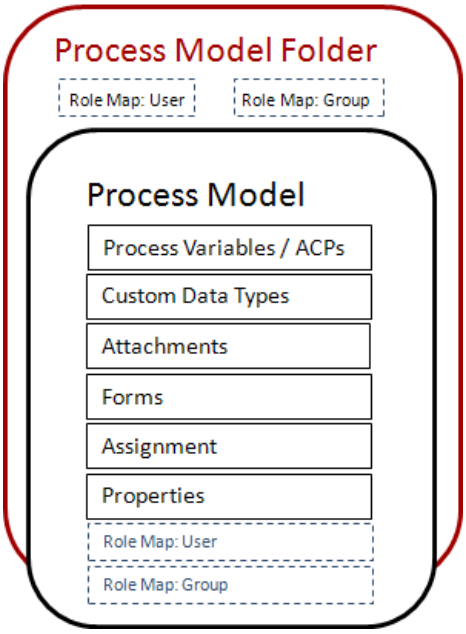
If a group type with an existing name (but different UUID) is imported, the import of the group type fails. Group type names must be unique within one installation (instance) of Appian.

The names of system group types cannot be updated.

If two or more attributes of a group type have the same name, only the first attribute is added to the group type. Group type attributes must have unique names.

Processes

Process models and process model folders can be exported for import onto another server. Process instances are not exported.



Process Model Folders

A process model folder export includes the folder's properties and user rights. The contents of a process model folder (any sub-folders or process models) must be selected individually for export as well, and are not exported automatically with the folder.

The My Models folder only exists on the server when it is *used* for the first time. This occurs when a subfolder is created within it, when a process model placed in it (or if the folder is renamed for any reason). Only after the folder has been *used* is it exportable. For the sake of import and export, the My Models folder is treated as any other process model folder, and is created at the root level (in the left navigation of the Processes view) when imported.

API DEVELOPERS ONLY: The ProcessModelFolder `type` and `isSpecial` attributes are not exported. On import, the `type` attribute is set to `TYPE_COMMUNITY`, if the folder has no parent. Process model folders that have a parent folder inherit the type of the parent folder.

Process Models

A process model resides in a process model folder. A process model export includes its properties (name, description, etc), definition (process flow), user rights, a reference to the process model folder where it resides, and a reference to any custom dashboards. A process model can include references to many other object types. A referenced object must be present on the target system - or you must include it in the application package - to successfully import the process model.

When exporting a process model, the latest published version is exported (if published).

- If no published version exists, the draft version is exported.
- References to other objects are determined based on the version of the process model that is exported.
- Process models exported from the Process Modeler (using an earlier version of Appian) cannot be imported as an application.

On import, process models are published as a new version.

- If you are reimporting an existing process model, draft versions you may be working on are overwritten when the imported process model is published.
- When importing process models (for the first time) that are linked recursively, the process models are published twice - resulting in two new versions.
- Subsequent imports (updates) publish the recursively linked process models one time per import, creating one new version.
- The user performing an import is listed as the creator of notes and attachments associated with the process model, as well as the owner of the process model.
- On import, if the process model references smart services that do not exist on the target system, the import of the process model fails.
    - The import behaves in the same way for missing referenced objects, such as a deleted CDT.
- The Process Modeler cannot import process models from application packages.
- Process models that are exposed as a web service must be re-exposed after import on the target system (if the process model is expected to be hosted on the target system).

Importing an application package that contains a process model makes it available to the Process Modeler, in a similar manner to the legacy process model import.

If a process model contains a Send Message event that specifies another process model for its destination configuration, and if the model generally targets a process model, rather than a specific Receive Message event in the process model, the process model (with the Send Message event) can be imported - but not published.

- In such cases, you must edit the process model to specify the target of the send message event, before it can be published.
- See also: Messaging Best Practices

If a process model contains a smart service that stores passwords, such as the Query Database and Call Web Service smart services, the passwords are not exported.

- On import, the process model is created but not published.

If a process model contains an Escalation Task that uses rules and/or constants in an expression for the task recipient, the rules and/or constants are not exported.

- On import, the process model is created but may produce an error and fail to publish.
- In such cases, you must manually add the rule and/or constant, before the process model can be published. Make sure to the name of the rule/constant is the same as what the expression references.

Custom Data Types

Process Models can use custom data types to define various objects such as process variables, node inputs, and data store entities.

When an import package contains a custom data type with the same name and namespace as a type that already exists, the new data type is compared with the old.

- If they are the same, no problems are logged. Objects that depend on the data type are allowed to be imported.
- Should the comparison indicate that the custom data type on the target system has a different structure than the one in the import package, the data type contained in the application package is **not imported**.
    - In such cases, a duplicate type definition problem is logged in the import log.

Objects in the package that use an inconsistent data type are still imported. These objects use the existing data type definition of the same name.

- It is possible for an inconsistent data type definition to cause unexpected issues at runtime.
- Inspect the existing type definition whenever a duplicate data type definition problem appears in the import log. It is important to determine whether the existing definition may or may not be compatible. If an incompatibility is found, the imported application should not be used until the custom data type is updated to match the exported data type.

CDT metadata is not imported. Imported custom data types take the following properties:

- `XSD Import` is listed as its **Origin**.
- The user who imports the application is listed as its **Creator**.
- The date the application is imported is listed as its **Date Created**.
- The visibility of a CDT remains the same as on the source system.

Application packages created using an Appian version that did not support exporting and importing CDTs record an error if the data type doesn't already exist on the target system.

- Objects that depend on legacy CDTs (migrated from earlier product versions) are created with the following exception.
    - References to any deleted (or missing) data types are modified to use the current definition of the data type that exists on the target system, even if the CDT structure is different.

- To ensure that the correct CDT is available and imported on the target system, delete and recreate data types that were migrated from prior versions of Appian.
    - Ensure that you have a configured primary data source to store the CDT when recreating the legacy types.

You **cannot** export a deleted CDT.

Custom data type definitions that are created from web service WSDLs are created using the following process.

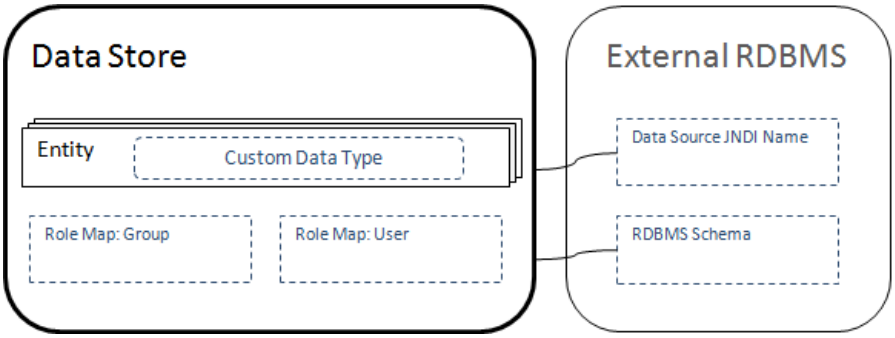
- Appian looks for the existence of the custom data type on the target system.
- When the data type doesn't already exist, Appian calls out to the WSDL to retrieve the type definitions.
- Appian looks at the data type definitions in the import package.
- If the WSDL is available and the data type structure does not match what's included in the import package, a data type definition warning appea in the import log.
    - Inspect the existing type definition whenever a structural data type definition warning appears in the import log. It is important to determir whether the existing definition may or may not be compatible. If an incompatibility is found, the imported application should not be used until the process is updated to match the current WSDL.

Best Practice: We recommend including any custom data types in your application package that would otherwise be created by the Call Web Service Smart Service activity during import.

- Use the Call Web Service activity in a development environment to initially create the types needed.
- Add these data types to your application, using the Data tab in the Application Builder.
- See also: Updating a Complex Data Type

Data Stores

Data stores can be added to your application packages for export and import.



Exporting Data Stores

When exporting a data store, the data source JNDI name is also exported.

- A data source with an identical JNDI name must exist on the target system, for successful import.
    - If Data Stores use a different JNDI name on the target system, the imported data store must be edited to update the data source used.
- The latest version of a data store (at the time the application package is created) is the one that is included.
    - Draft (unpublished) data stores can be exported, if they are not yet published.
    - Once published, only the latest published version can be exported.
- Related object references for a Data Store are determined based on the current version being exported.

Importing Data Stores

- Custom data types used by data store entities must be included in your application package for the import to be successful, or the data type mu already exist on the target system.
- If a version of the data store already exists on the target system, the existing database schema and entities are verified, but are not automatically updated.
    - Any missing database tables are not automatically created.
    - Any existing tables are not modified.
- If the datasource or entity mappings aren't valid, the data store is created (or updated) as an unpublished new version or draft. This allows you t address any issues (such as updating the datasource) while retaining any references to other application components.
- Any validation errors are listed in the import log. The cause of the error is also listed.

Expected Results when Importing Published Data Stores:

The following results occur when importing an application package that contains a published data store.

- The import process attempts to publish a new version of the data store, when the package contains a published data store.

Data Store Version on Target System	Result when Publishing Succeeds	Result if Publishing Fails
None	A new published version is created.	• A new draft version is created. • Imported objects that reference the data store retain their references.
Draft	A new published version is created. The draft	• The draft version on the target system is updated with the draft version in the application package.

Draft	version is discarded.	• Imported objects that reference the data store retain their references.
Published, without a subsequent draft version.	A new published version is created.	• A new draft version is created. • The published version is not affected. • Imported objects that reference the data store retain their references.
Published, with a new draft version.	A new published version is created. The target system draft is discarded.	• The draft on target system is updated with the draft in package. • Imported objects that reference the data store retain their references.

Expected Results when Importing Draft Data Stores:

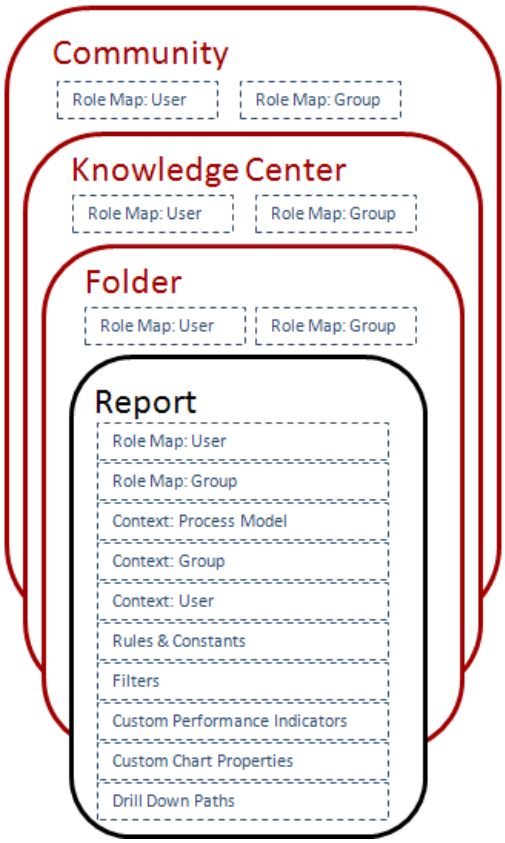
The following results occur when importing an application package that contains a draft (unpublished) data store.

Data Store Version on Target System	Result on Import
None	A new draft version is created.
Draft	The draft on the target system is overwritten by the draft in the application package.
Published, without a subsequent draft version.	A new draft is created; the published version is not affected.
Published, with draft	The draft version in place is overwritten using the draft in the application package.

Reports

Reports are managed as documents for export and import. References are retained by reports after export and import, except for certain contexts. Supported references are listed when using the Missing Dependencies tool in the Application Builder.

See also: Missing Dependencies



The following references are retained on import when included in an application package.

- Report contexts that reference process models, groups, or users
- Rules and constants in column definitions and drilldown paths
- Drilldown paths to Appian objects such as another report or dashboard
- Filters
- Custom indicator icons
- Custom chart properties

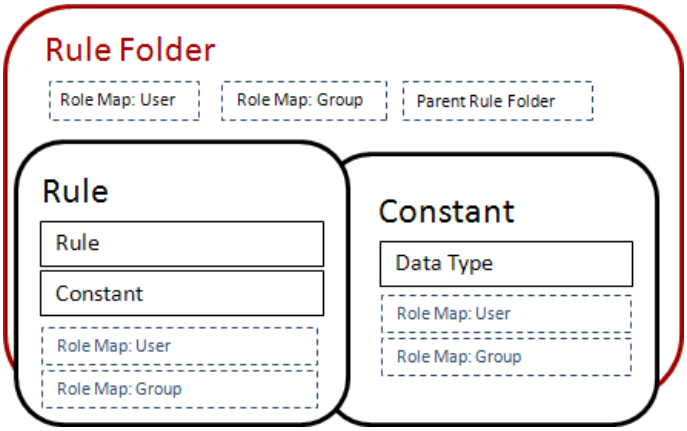
Reports exported prior to Appian 6.0.2 do not include these references. Export the report again using an upgraded version of Appian to enable the references.

Process Instance Context

When a report definition specifies a process instance context, the context will not be preserved after the report is exported. (Individual process instance cannot be exported or imported.) The report context must be reconfigured after import. This does not impact reports without a context, or those that take a run-time context (from a dashboard when shown in a Report Channel, or by prompting the user to select a context).

Rules

Rules folders, rules, and constants can be exported and imported onto another server.



Rules Folders

The Rules folder export includes its properties, user rights and a reference to any parent folder. The contents of a rules folder (sub-folders, rules, and constants) are not exported automatically. The rule folder must be included in your application in order to import a rule or constant.

Rules and Constants

A rule or constant export includes its properties, definitions, user rights, and a reference to its parent folder. A rule definition can reference other rules ar constants. A constant can reference most other object types. When exporting a rule or constant, the latest version is exported. On import, a new version of the rule or constant is created.

Duplicate names for rules and constants are not allowed. On import, if the name of the importing rule or constant matches an existing rule or constant, or matches an existing expression function, the import fails.

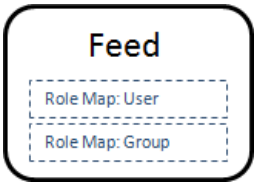
Updates to the names of rules and constants are not allowed. The import fails if the name in the package does not match the one in the target system (identified by UUID).

- Rules and Constants XML files exported from the Rules interface are not importable from the Applications view. Conversely, you cannot import a rule in an Applications package into the Rules view.

Feeds

When exporting feeds, the feed properties and user rights are included.

- User subscriptions to feeds are not changed if a feed is updated.
- Individual feed postings and comments cannot be exported.
- When a feed is updated on import, its postings and comments are not updated or changed.



System Objects

System objects are created automatically by Appian, and can be included in your application packages for export. Examples of system objects include system reports, the System Knowledge Center, and other objects such as the default process dashboard page.

System objects can be exported and imported. Each system object is given a predefined UUID (also called a system UUID) with the format `SYSTEM_<OBJECT-TYPE>_<OBJECT-NAME>` that is the same on each installation of Appian. This means that on import, the corresponding system objec on the target server is updated.

The following system objects are not listed as application dependencies when scanning an application using the Missing Dependencies tool.

- Group Type Departments
- Group Type Teams
- Group Type Custom
- Community Root

- Default Community
- System Knowledge Center
- Email Template Folder
- Leader Message Pictures Folder
- Priority Icons Folder
- System Reports Folder
- System Reports
- User Pictures Folder
- Temporary Docs KC
- Temporary Docs Folder
- Green Report Indicator Icon
- Red Report Indicator Icon
- Yellow Report Indicator Icon

If you modify one of these system objects, and you want to include it in your application, you can select it in the same manner as other objects of that same category.

See also: Application Builder

Notes for Application Administrators

If you are moving complete applications from a staging environment to a production environment, take standard precautions to ensure continuity.

- Before importing an application to your production site, create a backup the engine files (KDBs) if possible.
- To create a backup of your production objects, export all of the existing objects associated with your application. If the import should fail for any reason, this allows you to restore the previous state of the application.
- Because an import is not one atomic operation, users may see some inconsistencies in their application behavior while the import is running. For example, during a lengthy import, a Constant might be updated, but the Process Model that references it has not been updated yet.
    - We recommend that production-environment application imports be performed during off-peak hours, especially when importing large applications.

Security

You have to have at least viewer rights for any object to export it.

The user who imports an object is listed as its creator, but is not given administrator rights for that object.

You must have the right to create an imported object in order for an import to succeed. For example, only a System Administrator has the rights to create a Public group (non-system administrators can only create Public groups within groups that they administrate).

If you are importing (updating) an object that already exists on the target system from a prior import, you must have administrative rights for the object, or the import fails. To update an object, you must have the right to update the object's properties, as well as the object's role map.

Assigning Administrator Rights to the User Who Imports an Application

The user who imports an application does not automatically have the necessary user rights to administer an application object. You must ensure that th importer of the application can administer it and create the objects that it contains.

Best Practice: For each application, we recommend using a group called _Administrators to hold the necessary object creation rights for successful import of your application.

1. Create the group on the source machine, and export it (in an application package) to the target machine prior to importing the main application.
2. Before exporting the application, add this group to all object role maps. Assign Administrator rights to this group.
3. Temporarily add the users (or user) who imports the application on the target machine to this group before performing an import.
4. After the import, remove users from the _Administrators group, as needed.

Your application administrators group must be uniquely identified as the same group on both the source and target systems. The group must be deployed in an application package in order for Appian to recognize it as the same group on both systems.

Deploying User Rights with Exported Objects

Almost all Appian objects are associated with a set of users and groups who hold certain rights to view, execute, or change the object. When objects a deployed to a new Appian installation, the associated user rights are also moved. The deployed objects contain a reference to the users and groups liste in their role maps. For example, an exported Knowledge Center contains the list of users and groups with rights to the Knowledge Center. A deployed object's role map (and resulting user rights) can vary based on the objects that are present in the import package and on the target server.

Role map deployment rules:

- A user or group is only included in the role map of a deployed object if at least one of the two conditions below is true.
    - The user or group is present in the import package but not on the target server.
    - The user or group exists on the target server (and the user is not deactivated). All other users and groups are dropped.
- When importing an object that already exists on the target server (identified by UUID), the resulting role map contains a union of the users/grou between what's on the target server and what's listed in the import package.
- When importing an object that already exists on the target server (identified by UUID) and the access level for a user/group on the target machine does not match, the imported object takes the access level listed in the import package.
- Some objects can be configured to inherit their user rights from their parent. This configuration is preserved when the object is exported and imported. If the configuration of the object differs in the import package from the target server, the setting in the import package is used.

User Rights Deployment Example

Export Package

An exported Knowledge Center has the following role map:

Role	User or Group
Administrators	group1, user1
Authors	
Readers	group2

Target Server

The same Knowledge Center and group2 exist on the target server, but group1 and user1 do not. The Knowledge Center on the target server has the following role map:

Role	User or Group
Administrators	group3
Authors	group2, group4
Readers	

Target Server after Deployment

After the package is imported onto the target server, the following role map for the Knowledge Center is created as a result:

Role	User or Group
Administrators	group1, group3
Authors	group4
Readers	group2

Deployment Results

After this example import has occurred, the following results appear in the role map.

- **user1**: Does not exist in the resulting role map.
- **group2**: Changes to a Reader from an Author because a conflict in access level favors the import package.
- **group1**: Exists in the role map because it was in the import package.