aPriori

aPriori Professional System Administration Guide

Version 2019 R2

Copyright

Document usage

Third parties

Software rights

Document Information

Last updated February 5, 2020

The latest version of this document can be found at the aPriori Support HelpCenter (requires registration): https://support.apriori.com/hc

CONTENTS

About This Guide

This section provides information about this System Administration Guide, and the other ways in which aPriori supports the aPriori application.

Key topics include:

- Overview
- Related documents
- Typographic conventions
- Note: Notes highlight information, provide supplementary information, offer efficient or easy ways to perform tasks, or explain how to prevent errors or data loss.
- Feedback and customer support

Overview

This *System Administration Guide* contains detailed information about administering the aPriori solution using the System Admin Toolset. It is designed to be used as a reference for aPriori system administrators.

Related documents

In addition to this guide, you can find more information about the aPriori application in the following documents:

- *aPriori User Guide* – This guide contains detailed information about the aPriori solution. It is designed as a reference for your everyday work.
- *aPriori Cost Model Guide* – This guide contains detailed information about process groups and includes a chapter on direct and indirect overhead. Note: This is a new document as of 2015 R1 SP1 and contains chapters that formerly appeared in the aPriori *User Guide*.
- *aPriori Cost Model Workbench User Guide* – This guide explains how to use aPriori's Cost Model Work Bench (CMWB) to customize cost models.
- *aPriori VPE Administration Guide* – This guide contains detailed information about using the tools in the virtual production environment (VPE) toolset to maintain the VPEs in your aPriori deployment. It is designed as a reference for VPE administrators.
- *Release Notes* – This document highlights the changes made in aPriori since the previous release. It also contains last minute information about the release.
- *Installation Guide* – This guide contains detailed information about installing aPriori.
- *System Requirements* – This document provides information on the minimum and recommended client and server requirements to run aPriori, as well as the CAD file formats supported by aPriori.

Typographic conventions

The following conventions are used in this guide to convey additional information.

Style	Application	Example
`Code`	Literal text. The text appears exactly as shown. Usage includes commands, directory names and paths, file names, and system information.	`E:\setup.exe`
`Italic code`	Values that you specify. For example, you need to supply a value for *`your file`* in the path name example to the right.	`C:\aPriori\`*`your_file`*

Style	Application	Example
GUI	Objects in the aPriori interface	The **Document** field …
GUI Action	Interaction objects in the aPriori interface	Click **OK**.

Note: Notes highlight information, provide supplementary information, offer efficient or easy ways to perform tasks, or explain how to prevent errors or data loss.

Feedback and customer support

We appreciate your comments about this guide. Please contact us with your comments, questions, and requests for technical support.

Website:	http://www.apriori.com/support
Email:	support@apriori.com

1 System Administration Overview

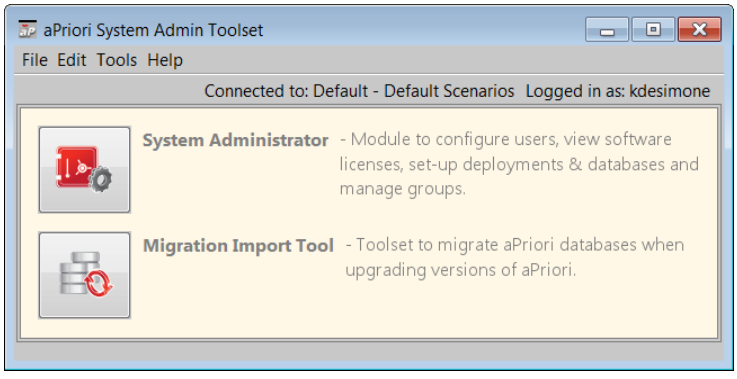
The System Admin Toolset can be accessed only by an aPriori system administrator. It allows you to perform traditional IT functions, such as administering users, licenses, and deployment configurations, as well as migrating data to new versions of aPriori.

The System Admin Toolset includes:

- **System Administrator** – Use this tool to add, edit, or delete aPriori users, view software licenses, set up deployments, set up databases, assign permissions to groups, create user defined cost object attributes, and customize the **Search** and **Cost Object Info** windows.
- **Migration Import Tool** – Use this tool to migrate aPriori databases when upgrading to a new version of aPriori.

To access the System Admin Toolset

Select **Tools > System Admin Toolset** from the aPriori menu bar to display the **aPriori System Admin Toolset** window.



2 System Administrator

The aPriori System Administrator tool allows System Administrators to manage users, licenses, deployments, groups, and user-defined cost object attributes, and customize the **Search** and **Cost Object Info** windows.

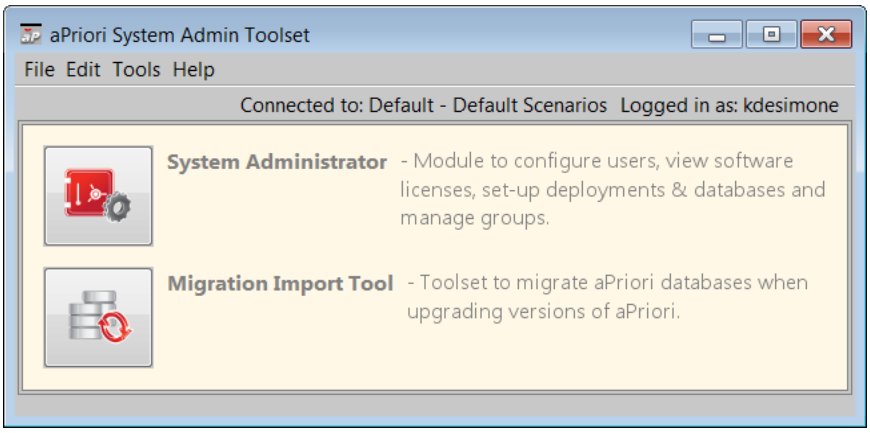
This chapter includes the following topics:

Overview

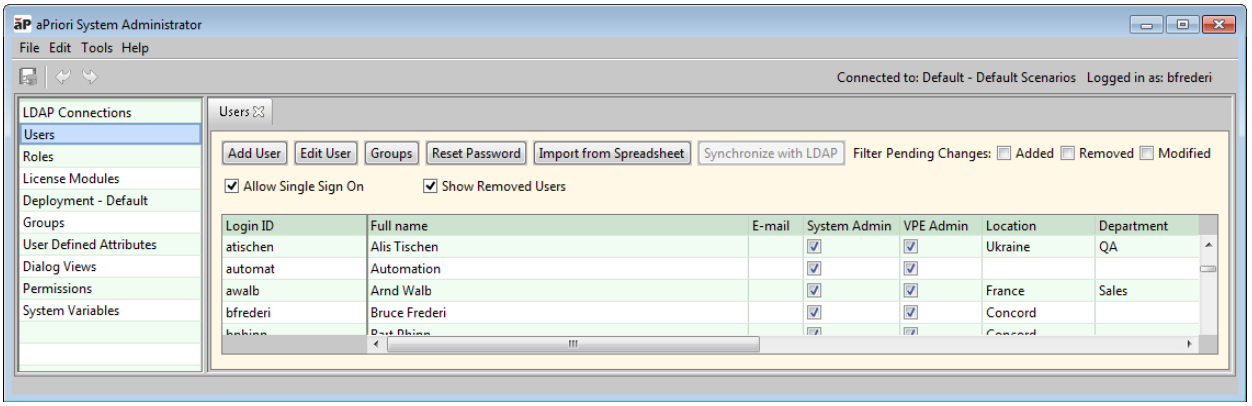
The aPriori System Administrator tool allows you to manually add, edit, or delete aPriori users, define LDAP connections for more automated user management, view software licenses, set up deployments, configure User Defined Attributes and their display, set up databases, configure Access Control through Groups and Permissions, and customize the Search and Cost windows.

To display the System Administrator window

1 Select **Tools > System Admin Toolset** from the aPriori menu bar to display the **aPriori System Admin Toolset** window.



2 Click **System Administrator** to display the aPriori System Administrator window.



Note: Some of the **Roles** and **System Variables** options are intended for Cost Insight Design, future expansion, and/or enhancements. You should NOT use these options unless explicitly directed to do so either by the documentation or by aPriori Support or Services representatives. Changes to these settings could significantly and negatively impact disk space usage and product performance.

Managing LDAP Server Connections

This section assumes that you have knowledge of general LDAP (Lightweight Directory Access Protocol) concepts and features.

The examples in this section use LDAP conventions. If you use Active Directory, adjust the examples accordingly.

The most basic user management approach in aPriori is adding, deleting, and modifying user accounts and their privileges and group membership manually through the System Administrator user interface. For this manual approach, see *Managing users* later in this chapter.

aPriori also provides the ability to specify one or more LDAP server connections to import users into aPriori and authenticate them each time they start aPriori. You can specify a number of details that will be applied automatically when users are imported, including:

- User settings and access conditions
- Which Access Control groups users should be assigned to (optional)
- Up to 10 user "Extra" fields that can be populated with LDAP server attributes

Once your LDAP connections are defined as described in this section, you can synchronize aPriori user accounts with the LDAP server(s) either manually (see *Adding User)*, or automatically through the separately-licensed Cost Insight Admin LDAP Synchronization ("LDAP Sync") module (see the *Cost Insight Admin Guide to LDAP Synchronization).*

In summary, you can manage your users in three ways:

- Manually: through the aP System Administrator UI.
- LDAP Map: by running LDAP connections manually through the **Synchronize with LDAP** button on the **Users** tab.
- LDAP Synchronization: using the separately licensed aPriori Cost Admin "LDAP Synchronization" module to run your LDAP connections automatically on a set schedule.

This section describes how to define LDAP connections, whether you synchronize them manually or through the LDAP Sync module.

Notes about LDAP Connections and Access Control Groups

As of Release 2017 R1, membership for all access control groups is managed independently of LDAP EXCEPT for the following special (system defined) groups:

- All Users
- System Administrators
- VPE Administrators

The All Users group is always modified by LDAP.

The Systems Administrators and VPE Administrators groups may or may not be modified by LDAP, depending on how these groups are defined and how the **Sync Admin Group Membership** options are defined for the LDAP connection.

The special Super User sub-group of the System Administrators group is ALWAYS managed manually.

For more information about Access Control groups, see *Groups* in the *Access Control* chapter.

LDAP Connection Behavior as of Release 2017 R1

As of Release 2017 R1, LDAP connections have been enhanced in the following ways:

- A new Group Membership Process automatically adds users to groups based on permissions associated with the group. This process runs at the end of every LDAP Synchronization job, and automatically whenever you Publish if any of your changes have touched the Users, Groups, or Permissions tabs.
- Attribute mapping has been extended to handle constants, manual entry, and mapping both LDAP organizational units and LDAP group names.
- A single LDAP connection is designed to gather all of the information needed to create the user and to synchronize the user's attribute data from LDAP. In addition, relevant organizational unit information and security group membership can be mapped to the user's attributes stored in aPriori. In this way the automated Group Membership Process can associate the user to the correct Access Control groups.
- The user's provenance can either be "Manual", an LDAP connection, or blank.
- Manual means that LDAP connections should not modify this user (changes are managed by admins through the UI).
- LDAP connection name means that the named connection will manage this user; other connections will ignore it.
- Blank should be a temporary value that is used for users that are intended to be managed by an LDAP connection. When the user is provisioned properly in LDAP, a sync will pick this user up. We do not want to put the connection name in since if the user is not yet provisioned in LDAP, a sync will not find the user and LDAP map/sync will remove the user from the aPriori database.

Note: When importing users from a spreadsheet, if a user's provenance is changed from LDAP to either blank or Manual, their password is reset to the default value provided during spreadsheet import. On their next login, they will be prompted with a **Reset Password** dialog (the same as a new user added via spreadsheet).

LDAP connections can be manually invoked one at a time through the UI ("LDAP Map"), or multiple connections can be run in an automated sync job ("LDAP Sync", which requires the separately licensed LDAP Synchronization module). When using LDAP Map, the administrator can invoke all or some of the connections and can process the additions, modifications, and deletions after each connection is synced. For an LDAP Sync job, the connections that are contained in the scheduled job will run and the changes will be published (no opportunity for manual massaging of the results).

When running LDAP Map, the user must "publish" the changes to the database by clicking on the **Publish Changes** icon in the toolbar..

The following rules describe connection logic during a sync (both manual and automated).

- User is returned by the sync and the user already exists
    - User's provenance matches the LDAP connection or the user's provenance is blank then the user is updated and all attributes (including the user's provenance if it is blank) get updated; user is added to the modify list.
- User is returned by the sync and the user does not already exist
    - The user is created, and all attributes are updated including setting the provenance; user is added to the addition list.
- User is returned by the sync and the user's provenance does not match the LDAP connection
    - The user is not updated.
- User is returned by the sync and the user's provenance is "Manual"
    - The user is ignored (not put on the modified list either).
- User is not returned but the user's provenance matches the LDAP connection
    - The user is marked for deletion (i.e. put on the deletion list).
- User is not returned, and the user's provenance does not match the LDAP connection, or it is **blank**
    - The user is ignored.

The following table summarizes this behavior in terms of the System and VPE Admin checkboxes on the **aPriori** tab of the LDAP Connection dialog box:

Conditions of users		
	same Provenance	**different Provenance**
Users returned by LDAP qry	group membership impact - see table below	ignored
Users NOT returned by LDAP qry	removed from aP	ignored

changes for users returned by LDAP query AND have matching or blank Provenance			
		Admin groups (sysAdmin & vpeAdmin)	
		checked	**unchecked**
Sync Admin Group Membership	**checked**	added to group (or remain in group)	removed from group if a member
	unchecked	no changes	no changes

The addition, modification and deletion lists are processed when the **Publish** button is clicked.

To change the connection that a user is provisioned by, an admin first changes the user's provenance field, clicks **Publish,** and then runs that newly-specified connection.

Multiple LDAP connections

The behavior of how LDAP Map or LDAP Sync handle multiple connections has not changed, except that they do not manage group membership anymore. In general, an individual user should be provisioned by a single connection. If a user is returned by more than once connection, LDAP sync will use the connection that matches the value of the user's Provenance field. If the provenance is blank, the provenance is set to the connection that has returned the user.

LDAP Authentication

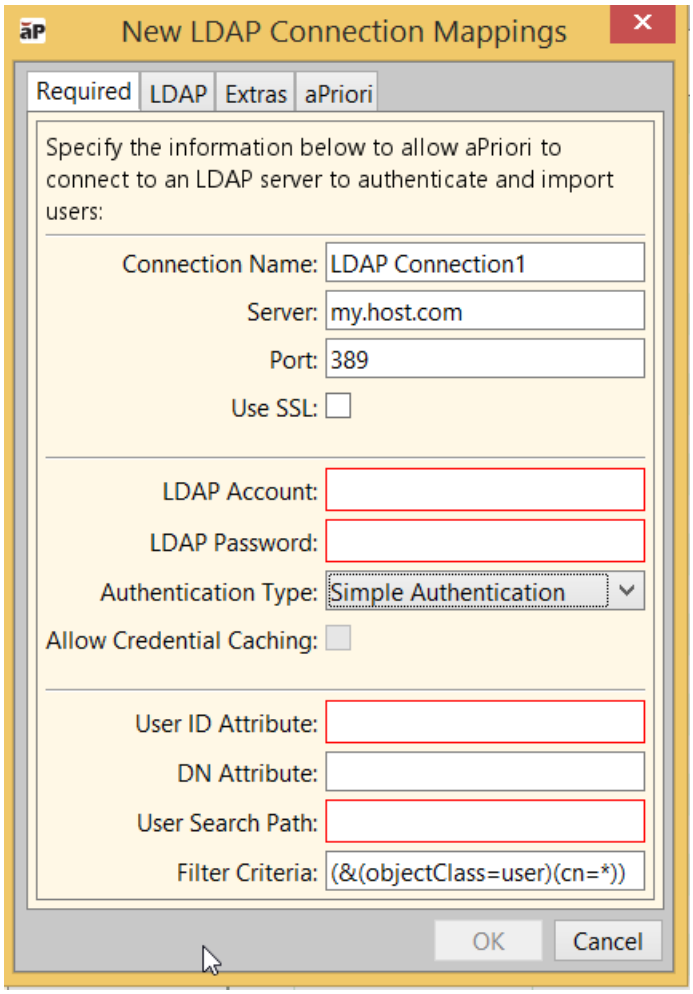
LDAP authentication is used when a user's provenance is set to an LDAP connection. You must first define the connection using the **aPriori System Administrator Window**. Connections can be defined as either **Simple Authentication** or **Kerberos**. If Kerberos authentication is used, you can elect to have Kerberos tickets cached, allowing automatic log ins to aPriori after an initial authentication.

Note: If SSO is turned on then the authentication via the user's Domain-cached Kerberos ticket will take precedence over the LDAP Kerberos cached ticket. In simpler terms, when SSO is enabled, the user's Windows authentication will take precedence over the LDAP connection authentication.

(See *Managing Single Sign On options* for information about Single Sign On authentication which also allows for automatic log ins.)

Simple Authentication

- Login: when aPriori is started it presents a login dialog and the user is asked to enter their username and password.
- Authentication: aPriori authenticates the user's credentials using the LDAP connection (i.e. the LDAP directory is used).



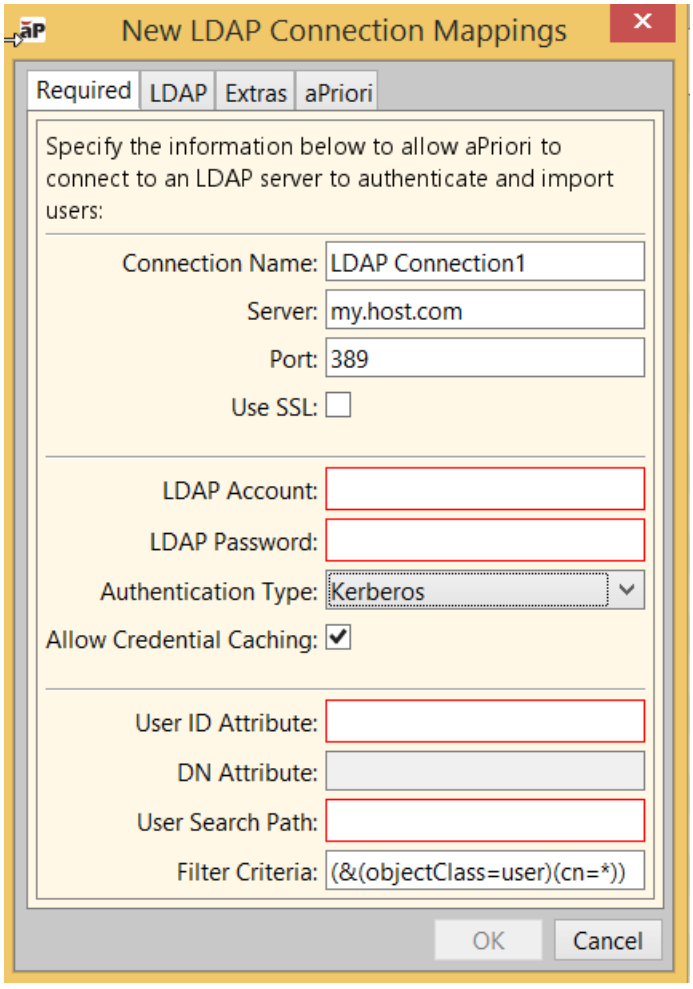
Kerberos (including "automatic log in")

When you choose Kerberos authentication, you can choose whether or not specify **Allow Credential Caching**:

Credential caching disabled: the login and authentication process works similar to **Simple Authentication** above.

Credential caching enabled ("automatic log in"):

- Login: when aPriori is started for the first time it presents a login dialog and the user is asked to enter their username and password. The user's Kerberos credentials are cached. When aPriori is started the next time, aPriori will skip the login screen and use the cached credentials (assuming the credentials have not expired and are still valid).
- Authentication: aPriori authenticates the user's credentials using the LDAP connection (i.e. the LDAP directory is used).

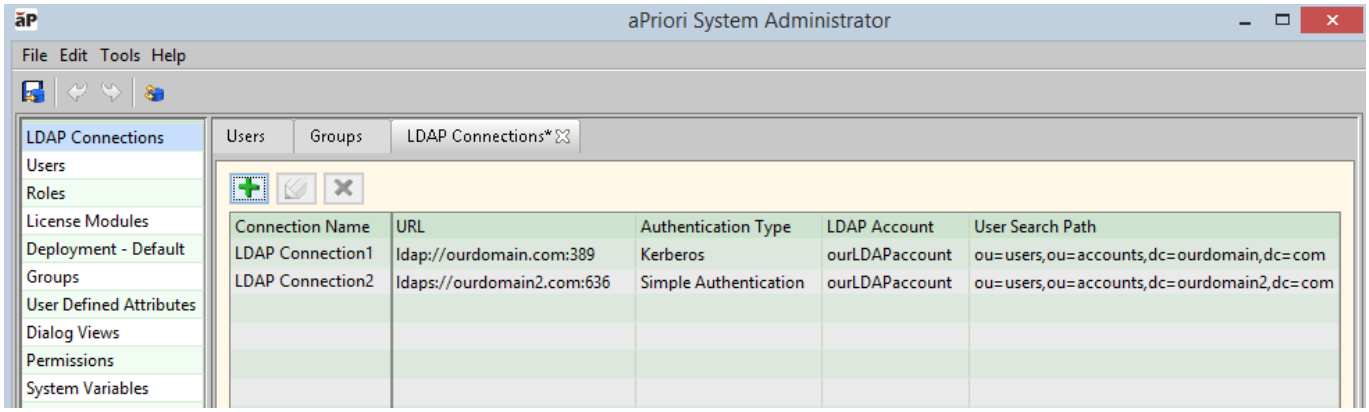


Defining an LDAP Connection

To display the LDAP Connections tab

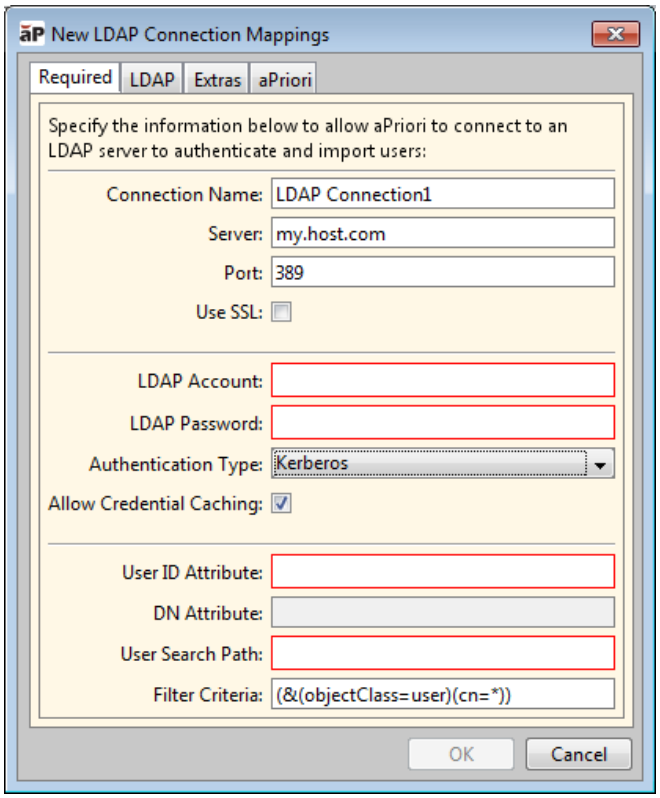
1 If the **System Administrator** window is not already open, click **Tools -> System Admin Toolset** from the main aPriori client.

2 The **LDAP Connections** tab should be displayed by default. If you are on another tab, click **LDAP Connections** in the Navigation pane to display it.



To add a new LDAP connection

1 In the **LDAP Connections** tab, click  to display the **Required** tab on the **New LDAP Connection** window.

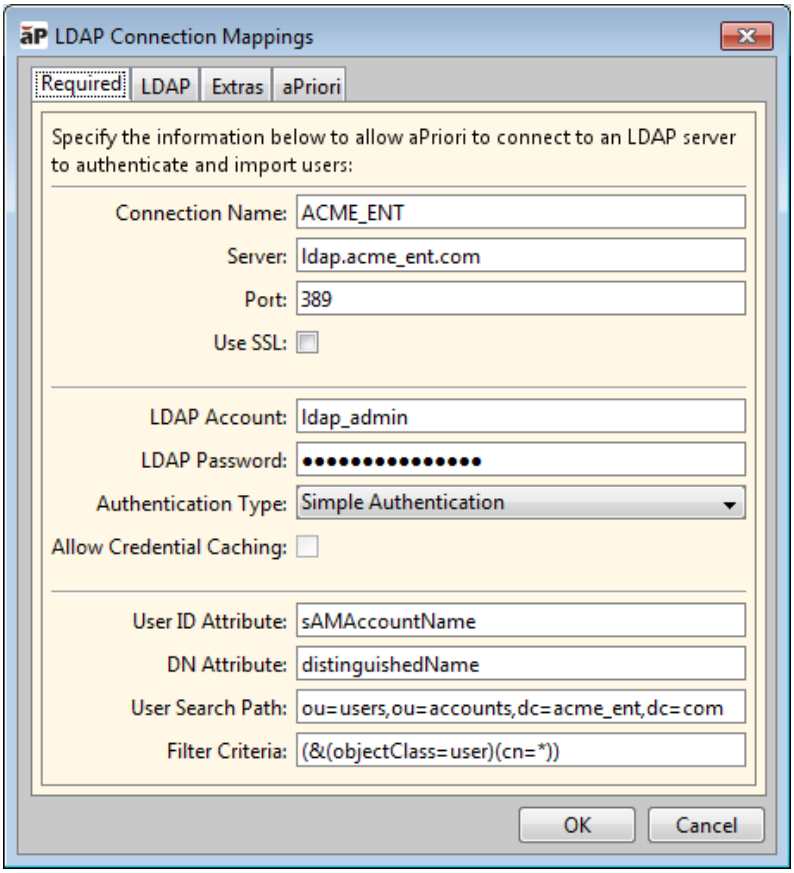


2 Complete the fields as described in the following table:

Column	Description
Connection Name	Enter any unique string used to identify the LDAP connection.
Server	Enter the LDAP server URL.
Port	Enter the port number used by the LDAP server. The default port is 389, which is not a secure SSL port.
Use SSL	aPriori recommends that for production work, you configure your LDAP server for SSL and specify a secure port such as 636. You need to configure the aPriori client to work with an SSL server. See *Configuring an LDAP SSL Connection* for more information. This checkbox changes the protocol from “ldap://” to “ldaps://”.
LDAP Account	Enter the user or account name that aPriori should use to connect to LDAP for user authentication and importing.
LDAP Password	Enter the password that aPriori should use to connect to LDAP for user authentication and importing.
Authentication Type	Select **Kerberos** (default) or **Simple Authorization**. **Note:** If you select **Kerberos**, and you encounter difficulty getting it to work, try placing a Kerberos configuration file in the aPriori JRE/lib/security folder or in the Windows system folder.
Allow Credential Caching	Check this box if you want aPriori to remember the user login credentials used to authenticate a user. This allows users to be logged into aPriori without entering a username and password if they are successfully authenticated. You must select the Kerberos authentication type to enable this option.
User ID Attribute	Specify the name of the LDAP attribute representing the user login ID. Typically "sAMAccountName".
DN Attribute	Enter the LDAP attribute that represents the Distinguished Name (typically "distinguishedName"). This field is not required if you are using LDAP for authentication only. It is ONLY required when an 'Org Unit' or 'Security Group' is selected as a Mapped Attribute on the Extras Field (described later in this chapter).
User Search Path	Specify the nodes of the LDAP tree structure that contain user objects. Valid entries must follow LDAP Search Path conventions and will depend on your LDAP implementation (for example, Microsoft Active directory vs. Open LDAP, etc.)
Filter Criteria	Optional. Enter criteria using LDAP Data Interchange Format (LDIF) syntax to limit which users are imported from the LDAP server. For more information, see *Filtering Imported Users* on page 16.

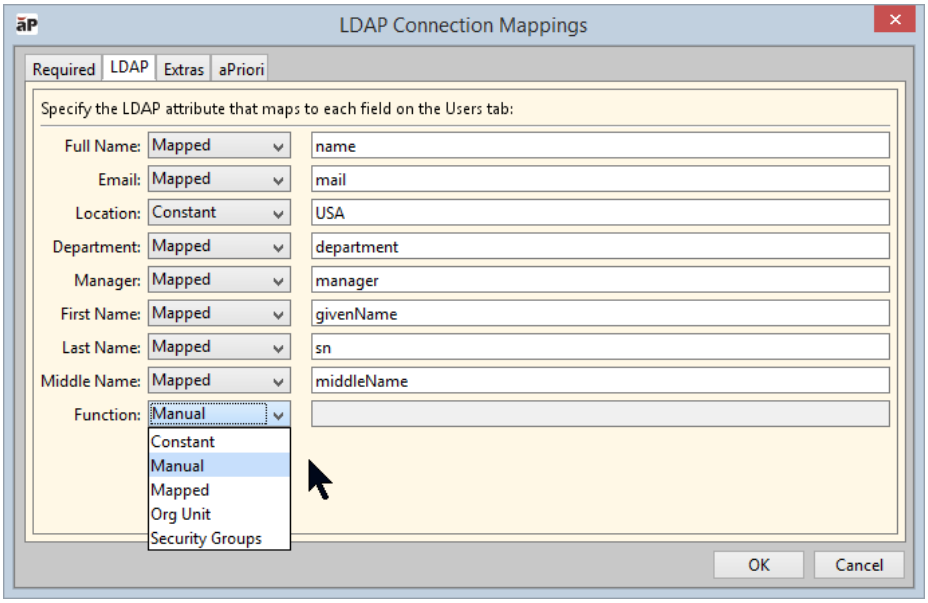
Column	Description
	Leave this field blank to import all the users in the specified user search path.

Depending on your company's details, the **Required** tab may look similar to this when completed:



Note: You must also provide settings for required fields on the **aPriori** tab (described below) before the **OK** button will be enabled.

3 (Optional) Select the **LDAP** tab to map LDAP attributes to fields in the aPriori user list. Common values are populated as defaults. For illustration purposes, the last field (**Function:**) is shown with an expanded drop-down menu.



When importing users from an LDAP connection, these attributes automatically populate the mapped fields in the aPriori user list, which can be viewed in the **System Administrator Users** tab and dialogs.

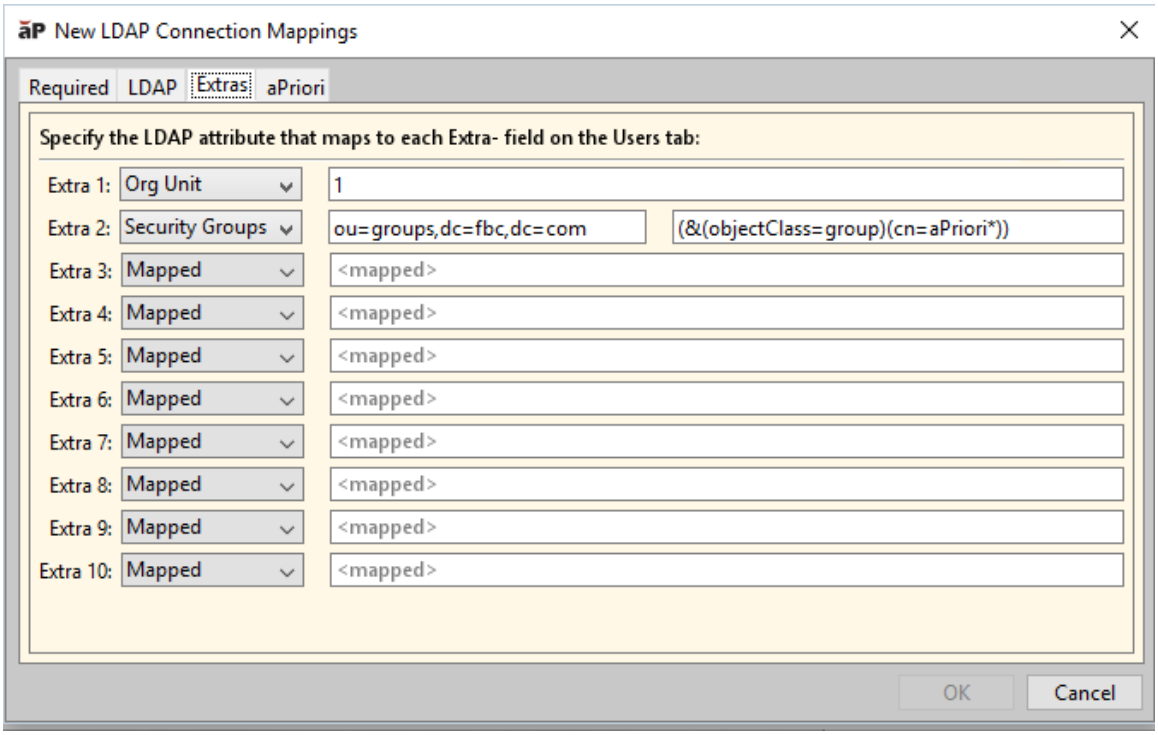
Note that specifying these attribute mappings is not required. If you do not map the attributes, then these aPriori fields will be empty and require manual edits to complete, if desired.

Here is a summary of the options available from the drop-down menu:

- **Constant –** Use this option when you want to set the field to a specific value. In the above screenshot example, the **Location** field has been set to a constant value of “USA” assuming that the LDAP connection is populating users from an LDAP repository that is location-specific.
- **Manual –** Use this option when you want the aPriori attribute to be filled-in manually by an administrator for each user, either from the UI or from a spreadsheet import.
- **Mapped –** Use this option when you want the aPriori user attribute to be set by the corresponding LDAP attribute. In the example screenshot above, the **Full Name** attribute field has been set to “name” so that it is populated by the LDAP “name” attribute.
- **Org Unit –** Use this option when you want to map an LDAP OU (Organizational Unit) to an aPriori attribute. For example, you might want to map the LDAP equivalent of the user’s Business Unit to one of aPriori’s “Extra” fields. The input control for this option is a level number that indicates where the Business Unit is located in the user’s DN (Distinguished Name path): the field prompt directs you to “**<enter OU level>**”. Note that there is no guarantee that each of the users being synced will have appropriate data in DN: for example, a user located on the 2nd level (a Vice President at the “company” level, for example) will have data for levels a and 2, but will not have data for level 3 (the “department” level, for example). Group membership will be determined based on the "member" group attribute which is (typically) not propagated in hierarchical group structure, so only groups where the user is a direct member will be recorded.

- **Security Groups –** Use this option when you want to store a list of security groups that the user belongs to. When this option is selected there are two input controls that need to be filled out. Note that there can be more than one group returned so the value in this field is a string that has a comma separated list of group names.
    - The first field specifies the **<group search path>** to the LDAP object that the groups are organized under.
    - The second field is the filter criteria for the groups of interest.

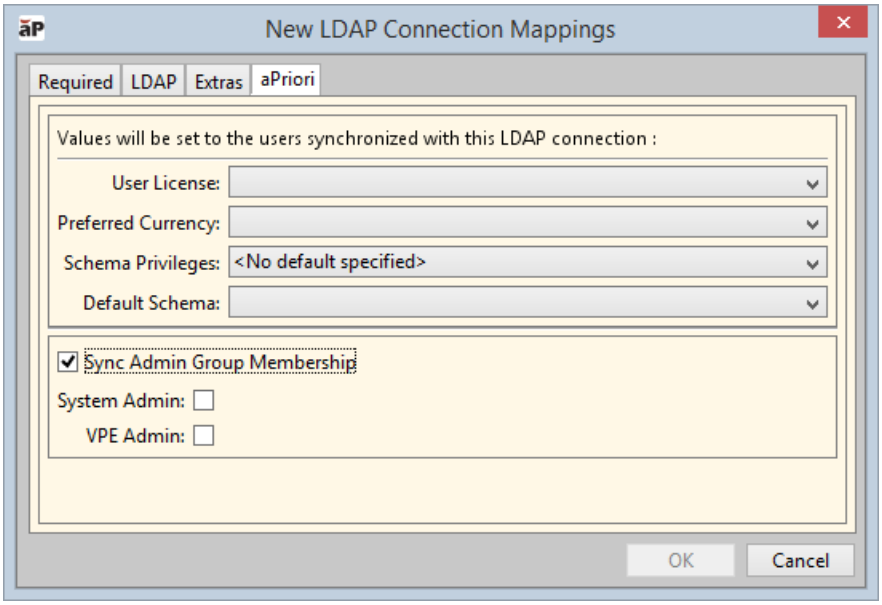
4 (Optional) If there are additional LDAP attributes that you would like to capture when users are imported, you can specify up to 10 on the **Extras** tab. In the following example, “Extra 1” extracts the top-level organizational unit (“OU”) out of the DN path, while “Extra 2” pulls all security group names starting with “aPriori” to which the user is a member:



These **Extra** user fields map to the **Extra** fields visible in the **Edit User** dialog box. Note that these user fields can also be populated from that dialog box, or by importing data from a spreadsheet, in addition to importing from LDAP attributes.)

5 (Required) You must also specify other values that are typically set on the **New User** dialog box for imported LDAP users, as well as automatically assign these users to specific aPriori Access Control groups. These fields are required so that any new users that are created during a LDAP Sync event have the necessary settings to log into aPriori.

Click the **aPriori** tab:



See the table under *Adding User* for details about the **User License**, **Preferred Currency**, **Schema Privileges**, and **Default Schema** settings.

Note: The following fields are required. The **OK** button will remain disabled until you provide values for these fields:

- **User License**
- **Schema Privileges**
- **Default Schema**

The **Sync Admin Group Membership** options provide for automated insertion of users into the System Admin and VPE Admin groups only. All other group membership must either be manually populated or managed through the Group Membership Process. Note that this is new behavior introduced in Release 2017 R1, to separate user definition and Access Control group membership.

The **Sync Admin Group Membership** checkbox enables the **System Admin** and **VPE Admin** checkboxes IF these groups are set to **Manual** in the **Groups** window (see *Groups Tab* in the Access Control chapter).

If these groups are set to **Automated** or **None**, the corresponding checkbox will be shaded, and you will not be able to change its state. If the checkbox is available, checking it will cause all LDAP users for this connection to be made members of that admin group (or to remain in that group if already a member). Leaving the box unchecked WILL CAUSE ANY LDAP USERS FOR THIS CONNECTION TO BE REMOVED FROM THAT ADMIN GROUP IF THEY BELONG TO IT. Note that if you do not have a Super User defined, this could cause you to lose ALL of your admin users if the LDAP connection includes all of your admin users, and you publish the results of the LDAP operation without first carefully examining those results.

6 When done, click **OK**.

If the **OK** button is disabled, check that you have completed all the necessary fields in the **Required** and **aPriori** tabs.

7 Select **File > Publish Changes** from the System Administrator menu bar or click in the toolbar to save your changes.

To edit an LDAP connection

1 Select **LDAP Connections** in the Navigation pane to display the **LDAP Connections** tab.

2 Select an LDAP connection and click to display the **LDAP Connection** window.

3 Edit the various tabs and fields as described in the previous section and click **OK**.

4 Select **File > Publish Changes** from the System Administrator menu bar or click in the toolbar to save your changes.

To delete an LDAP connection

1 Select **LDAP Connections** in the Navigation pane to display the **LDAP Connections** tab.

2 Select an LDAP connection and click to display the **Confirm Delete** window.

3 Click **OK** to delete the selected LDAP connection.

4 Select **File > Publish Changes** from the System Administrator menu bar or click in the toolbar to save your changes.

A warning is displayed if the deleted LDAP connection is referenced by a user's Provenance field. If you receive this warning, update the user's Provenance field to a valid LDAP connection.

Filtering Imported Users

You can use LDAP Data Interchange Format (LDIF) syntax to limit which users are imported from the LDAP server. The search filter syntax is basically a logical expression in prefix notation (that is, the logical operator appears before its arguments).

Each criterion in the filter is composed of an attribute identifier and either an attribute value or symbols denoting the attribute value. For example, the criterion `(sn=Geisel)` means that the `sn` attribute must have the attribute value `Geisel` and the criterion `(mail=*)` indicates that the `mail` attribute must be present.

Each criterion must be enclosed within a set of parentheses.

Criteria use logical operators to create logical expressions. Each logical expression can be further composed of other items that themselves are logical expressions, such as:

```
(| (& (sn=Geisel) (mail=*)) (sn=L*))
```

This example requests entries that have both a `sn` attribute of `Geisel` and a `mail` attribute of any value – or entries whose `sn` attribute begins with the letter `L`.

Note: Active Directory does not support wild cards (`*`) in filter criteria for attributes of type Distinguished Name (DN) (such as `memberOf`, `member`, and `distinguishedName)`. Do not use wild cards in filter criteria for such attributes; reference the fully-qualified DN name instead.

The following table lists the logical operators that can be used to create filter criteria.

Logical operator	Description
&	Conjunction (and). All the items in the list must be true.
|	Disjunction (or). One or more alternatives must be true.
!	Negation (not). The item being negated must not be true.
=	Equality according to the matching rule of the attribute.
~=	Approximate equality according to the matching rule of the attribute.
>=	Greater than (according to the matching rule of the attribute).
<=	Less than (according to the matching rule of the attribute).
=*	Presence. The item must have the attribute, but the value of the attribute is irrelevant.
*	Wildcard. Zero or more characters can occur in that position. Used when specifying an attribute value to match.
\	Escape. For escaping characters such as *, (, or) when they occur inside an attribute value.

Configuring an LDAP SSL Connection

aPriori recommends that for security, any production environment LDAP installation be configured to use SSL (Secure Sockets Layer).

Note: For simplicity, we use the term “SSL” to refer to both SSL and TLS (Transport Layer Security). Over the past several years, TLS has replaced SSL, but many people are still more familiar with the term SSL.

LDAP server and general SSL configuration is beyond the scope of this documentation and must be set up by your IT department. However, once an SSL LDAP server is configured, you must configure your aPriori installation to be able to communicate with it.

This consists of adding the certificate of the issuing CA (Certificate Authority) for your certificate into the Java ‘cacerts’ keystore included with Java as part of your aPriori installation. The ‘cacerts’ file is a list of Certificate Authorities (CAs) trusted by Java. When establishing a connection using an SSL certificate, Java must be able to find a path from one of the trusted roots to your certificate. If there is no path found, Java will not establish the SSL connection. Get a copy of the root CA certificate (and if applicable, Intermediate) that issued the SSL certificate used by your LDAP server from your IT department to add to the aPriori SSL keystore. This might be named almost anything but should be similar to “certificate.crt” or “certificate.cer”. These instructions assume the certificate is in PEM format (You may also see this called a BASE64 encoded X.509 certificate).

If you open your certificate file in a text editor, it should look like this if it is in PEM format:

```
-----BEGIN CERTIFICATE-----
MIIDmzCCAoOgAwIBAgIJANlzcMphFdASMA0GCSqGSIb3DQEBCwUAMGQxCzAJBgNV
... <many more lines that look similar to the above> ...
pcCZfoaYop2djiciry7C
-----END CERTIFICATE-----
```

1 Place this file in a known location so the command line later in this procedure can find it.

5 Bring up a command window and navigate to:

`<apriori_install>\WIN64\jdk\bin`

6 Enter the following command:

```
keytool -importcert -file <certificatepath>\<certificatename> -alias
   <aliasname>
   -keystore <truststore> -storepass <password> -trustcacerts -
   noprompt
```

where:

<certificatepath>\<certificatename> – are the location and the file name of the certificate you copied in Step 1. For example, `c:\temp\server3_cert.crt`.

<aliasname> – is an arbitrary, unique name you assign to this certificate to identify it in the store. For example, `yourdomain_server3`.

<truststore> – is the truststore in your aPriori installation:

`<apriori_install>\WIN64\JDK\jre\lib\security\cacerts`

<password> – is the password for your keystore. The default is “changeit”. Hopefully, you or your IT department already has.

Assuming that the command executes with no errors, restart aPriori and test that LDAP connections using an SSL port works. If it does not, examine the aPriori log file for messages that can help you troubleshoot any problems.

Note: If you receive an exception that contains the text “`unable to find valid certification path to requested target`”, this generally means that there is a trust issue. If your organization has an internal Public Key Infrastructure (PKI), certificates issued by your internal Certificate Authority (CA) will not automatically be trusted by Java. Even if you use a third-party CA, it may not always be included with Java. Either way, the steps for adding the certificates are the same. Most CAs provide you with the CA public root and any intermediate certificates when issuing your certificate.

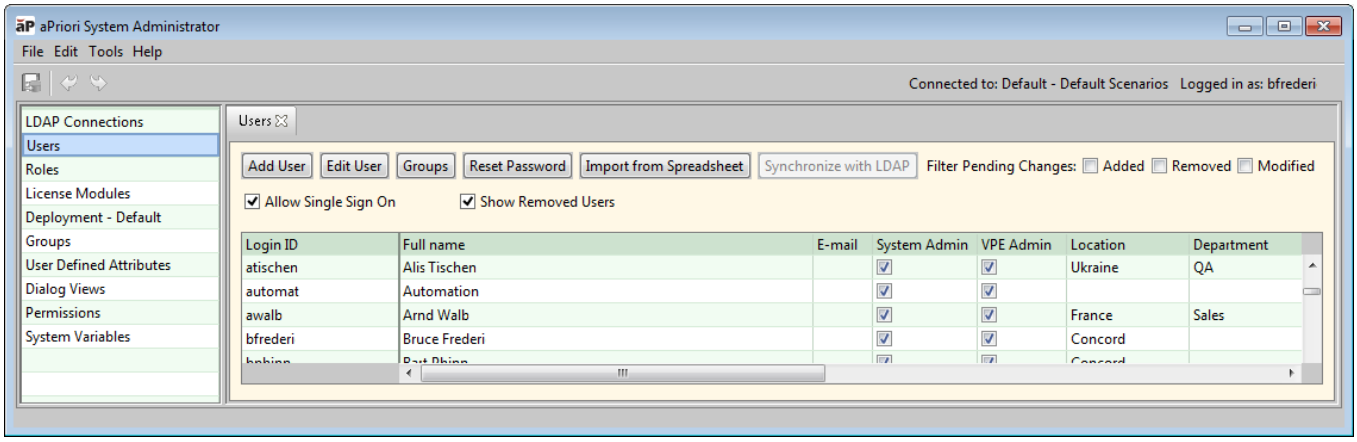
Managing users

The **aPriori System Administrator** window allows you to create, update, delete, and (re)activate users, as well as to control their access to the administration tools, set their default currency, and define their default component database.

Note: As of 2018 R3 SP1, all user change management actions (create, update, delete, and activate) are tracked and recorded in the aPriori database. This includes changes implemented through the UI, spreadsheet import of users, and ldap map/sync. If you activate a removed user and at the same time modify other fields, only “activate” is recorded.

To manage aPriori users

Click **Users** in the Navigation pane to display the **Users** tab.



Click a column header to sort the table by that column. Click again to reverse the sort.

Note: For information about the **Allow Single Sign On** checkbox, see .

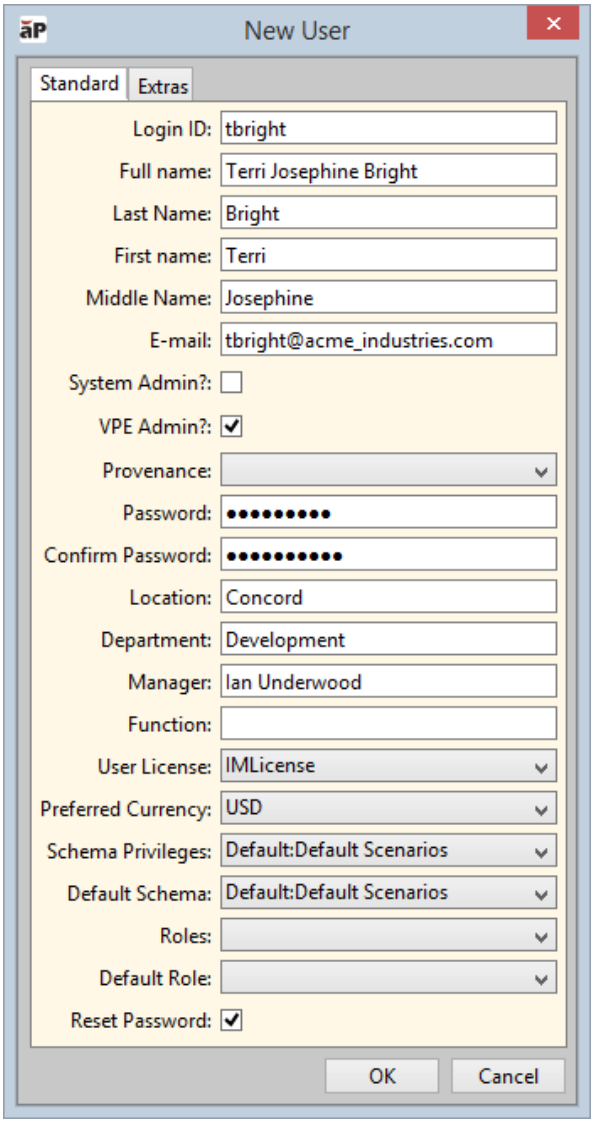
Adding Users

You can add users individually, or you can import multiple users simultaneously from a comma-separated values (CSV) file or LDAP server connection.

Users who are associated with an LDAP server connection are authenticated against LDAP each time they start aPriori.

Adding a Single New User

1 Click **Add User** on the **Users** tab to display the **Add User** window.



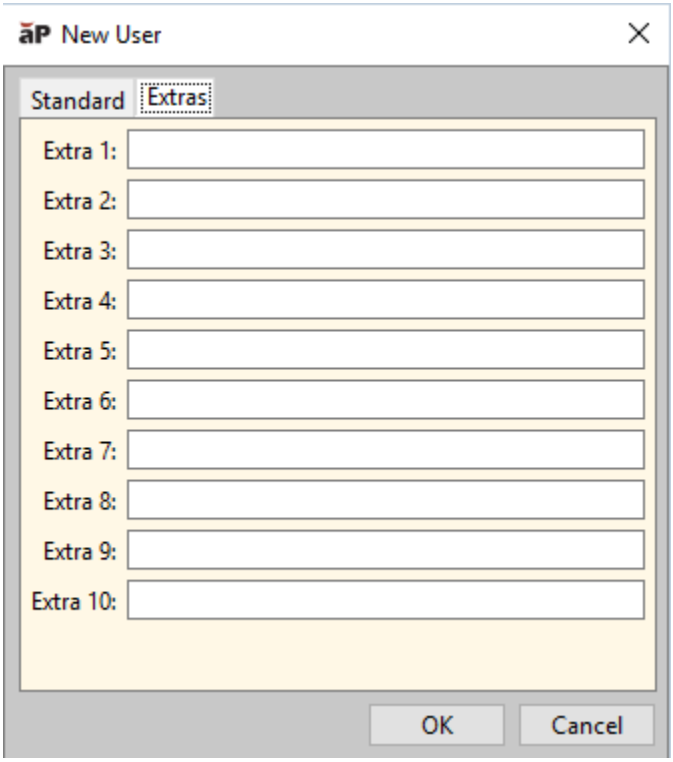
By default, the **Standard** tab is displayed.

2 Enter user information into the fields on the **Standard** tab.

Column	Description
Login ID	Identifier required to log in to aPriori.
Full name	Enter the full name of the user for the associated Login ID.
Last name	Last name of the user for the associated Login ID.
First name	First name of the user for the associated Login ID.
Middle name	Middle name of the user for the associated Login ID.

Column	Description
E-mail	The user's email address.
System Admin	Check this box to provide access to the System Admin Toolset. Uncheck the box to restrict access.
VPE Admin	Check this box to provide access to the VPE Toolset. Uncheck the box to restrict access.
Provenance	LDAP connection name. If your site uses one or more LDAP servers and you have defined LDAP connections for doing so, use this field to select the LDAP server connection against which the user will be authenticated. If you select an LDAP server connection, you will not assign the user a password. Select the blank option if you do not want to use LDAP authentication. **NOTE:** See *LDAP Connection Behavior as of Release 2017 R1* for additional behavior introduced in Release 2017 R1.
Password	Enter the user's password. If this field is disabled, LDAP authentication is selected.
Confirm Password	Re-enter the user's password. Once confirmed, the password is stored in encrypted format.
Location	The user's physical location.
Department	The user's department.
Manager	The name of the user's manager.
Function	Some companies specify a role or other job description for employees. Use this field to contain this kind information.
User License	Select the license ID (as provided by aPriori) that this user should be using.
Preferred Currency	Select the default currency in which component costs are displayed by default for the user.
Schema Privileges	Select the component database to which the user is connected by default when aPriori starts. (For information about deployments and schemas, see Editing User Settings in the Managing deployments).
Default Schema	If your company has multiple schemas to restrict access to specific cost objects, select the schema the user will be connected to by default.
Roles	Reserved for Cost Insight Design.
Default Role	Reserved for Cost Insight Design.
Reset Password	When set, the new user can log in one time with the password you have specified but must immediately change to a new password of their choice.

3 (Optional) Click the **Extras** tab if you want to define up to ten **Extra** user fields.



These extra fields enable you to use additional information about each user to affect aPriori behavior. You can use these fields in whatever way makes sense for your company. For example, you could use these in conjunction with Access Control rules (as properties to the "currentUser" subject), or to implement new custom workflow capabilities delivered by aPriori's Professional Services team.

4 Click **OK**.

The new user's Login ID is displayed in green at the bottom of the user list.

5 Select **File > Publish Changes** from the System Administrator menu bar or click in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes.

Importing Multiple Users from a CSV File

You can import users from a spreadsheet in the form of a CSV (comma-separated values) file.

In addition to the details listed in the procedure steps below, the following rules determine how entries are handled:

- Users are added or deleted as follows:
  - If a login id exists in the spreadsheet but not in aPriori, the user is added.
  - If a login id does not exist in the spreadsheet but does in aPriori, the user is deleted.
  - The last super user cannot be deleted
  - A currently logged-in user cannot be deleted.
- The provenance value from the spreadsheet overwrites any existing provenance value for an existing user. If a user's provenance changes from LDAP to either blank or Manual, their password is reset to the default value provided during spreadsheet import. On their next login, they will be prompted with a Reset Password dialog (the same as a new user added via spreadsheet).
- User attribute values overwrite any existing attribute values for a matching user (i.e., the spreadsheet cell value replaces the aPriori user attribute value based on column name).

Use the following procedure to import users from a CSV file:

1 Ensure that your CSV file contains the column headers you need from the list below so that aPriori can map data in those columns to specific fields in the aPriori users list. Only the loginID column is required. The other columns are optional.

- `loginID`
- `fullName`
- `firstName`
- `lastName`
- `middleName`
- `email`
- `isAdmin` – Use values of yes or no; true or false
- `isVPEAdmin` – Use values of yes or no; true or false
- `location`
- `department`
- `manager`
- `function`
- `schemaPrivileges`

  IMPORTANT: schemaPrivileges values must be entered in the format *<deploymentName>:<schemaName>*
  Also, If the SchemaPrivileges column has multiple values separated

by commas, then these should be enclosed in quotes so that the values are not imported into incorrect columns:
`"Default:Default Scenarios,Default:Development Scenarios"`

- `defaultSchema`

  IMPORTANT: defaultSchema values must be entered in the format *<deploymentName>:<schemaName>*

- `userLicenseName`
- `preferredCurrency` – Three-letter currency code. Possible values are:
  - `USD` – U.S. Dollars
  - `BRL` – Brazilian Real
  - `CAD` – Canadian Dollar
  - `CNY` – Chinese Renminbi Yuan
  - `EUR` – Euro
  - `GBP` – British Pound Sterling
  - `HKD` – Hong Kong Dollar
  - `INR` – Indian Rupee
  - `JPY` – Japanese Yen
  - `KRW` – South Korean Won
  - `MXN` – Mexican Peso
  - `TWD` – Taiwanese NT Dollar
- `roles`
- `default role`
- `provenance`
- `extra1` through `extra10` – See the previous section ("To add a new single user")

The currency code in your CSV file must also be active in the **Deployment Data** tool of the **VPE Toolset** to successfully set the user's preference. For more information, see the *aPriori VPE Administration Guide*.

Note: If existing users are not present in an imported spreadsheet, those users are removed, regardless of provenance setting.

A sample CSV file (`aPriori_Import_Users.csv`) is provided in the `install-support/templates` folder in your aPriori installation folder.

2 Click **Import from Spreadsheet** to display the **Select spreadsheet file** window.

3 Navigate to the CSV file, select it, and click **Open** to compare the users in the CSV file to the users already in the aPriori user database and display the **Synchronize with LDAP** window that summarizes pending changes from this comparison.

The **Import from Spreadsheet** window displays the:

- **Added users** – Number of pending new users; that is, users found in the CSV file that were not in the aPriori user database
- **Deleted users** – Number of pending deleted users; that is, users in the aPriori user database that were not in the CSV file

4 Click **OK** to display the following changes on the **Users** tab:

- New users' login IDs are displayed in green.
- Users to be removed are displayed in gray, with a strikethrough.
- Modified users' login IDs are displayed in blue.

5 Review your changes.

We recommend using the **Show Pending Changes** checkboxes at the top of the **Users** tab to quickly review your changes. Check the **Added**, **Removed**, or **Modified** boxes to show only those users.

Uncheck the boxes to show all users.

NOTE: Ensure that mandatory fields **Schema Privileges** and **Default Schema** are populated for each user. If not, ensure that the values in the .csv spreadsheet ar defined correctly (i.e., in the format `<deploymentName> : <schemaName>`. Proceeding to publish your changes without these fields correctly defined can result in the following error message when these users attempt to login: "Not authorized to connect to any schemas. Please contact your administrator."

6 Revert any pending changes by right-clicking one or more rows in the table and selecting one of the following options from the context menu:

- **Remove** – Remove the selected users from the user list
- **Undo Remove** – Restore the selected users marked for removal
- **Undo Edits** – Remove any changes made to a user via import or manual edits

Ctrl+click or Shift+click to select multiple rows.

7 Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes.

Change indicators are removed.

If you delete users, any aPriori licenses allocated to these users are released and available for assignment to other users.

Importing a User CSV File from the Command Line

As of 2018 R3 SP1, you can import a CSV user spreadsheet from the command line as well as from the UI (as described in the previous section). This command line approach is similar to aPriori BOM Loader and Bulk Costing functionality, as well as the Access Control group functionality also introduced in 2018 R3 SP1.

The command to run this functionality is:

```
<apriori_install>\bin\loadUsers.cmd
```

loadUsers.cmd syntax

```
loadUsers.cmd <importFilePath> <defaultPassword>
   {<userName>} {<userPassword>}
```

where:

<importFilePath> is the full path to the import user spreadsheet file.

<defaultPassword> is the password for newly created users.

<userName> is the aPriori user account under which the changes will be made.

<userPassword> is the password for the aPriori user.

<userName> and *<userPassword>* are optional if Single Sign On is enabled. The loader will try to log in using the current Windows user login if they are omitted. Note that the user should be a member of the Super Users group.

IMPORTANT: Although importing the user spreadsheet from the command line has the same basic behavior as importing through the UI, you will not have the opportunity to review the results before publishing the changes. This means that *if an active user is not included in the spreadsheet that user will be removed upon import.* So, for example, if you imported a small spreadsheet with just a few users for whom you wanted to make some changes, all the other users would be immediately removed upon import. The spreadsheet should include all active users and the way they should exist in the database after the import, even if they are not being changed. Here is a summary of all import behavior:

1. Users who are in the spreadsheet but not in the user database will be added and activated so long as all their details are valid.
2. Active users who are in the spreadsheet will either be:
   a. Modified if any of their fields have changed.
   b. Skipped if their fields have not changed. (“Skipped” means they are neither updated nor removed.)
3. Active users who are not in the spreadsheet **will be removed**.
4. Removed users who are in the spreadsheet will be activated if a valid license is specified for them. Other valid changes within the same row will be accepted. If you have an activate row and one or more rows that change other fields for that user, only the activate row will be accepted and the other rows will be skipped as duplicates. You cannot modify a removed user (except by activating them). Removed users who should remain removed do not need to be included in the spreadsheet.

Import Multiple Users by Synchronizing with an LDAP Server Connection

1 Configure one or more LDAP connections.

For more information, see *Managing LDAP Server Connections* on page 4.

2 Click **Synchronize with LDAP** to display the **Synchronize with LDAP** window.

3 Select one or more LDAP connections and click **OK** to compare the users from the LDAP connection's User Search Path (with filter criteria) to the users already in the aPriori user database and display a summary of pending changes from this comparison.

You can check the **Select/Deselect All** box to select all the LDAP connections. Uncheck the box to deselect all the LDAP connections.

The summary displays the:

- **Added users** – Number of pending new users; that is, users found in LDAP that were not in the aPriori user list
- **Deleted users** – Number of pending deleted users; that is, users in the aPriori user list with the Provenance set to the selected LDAP connection, that were no longer in the LDAP connection

4 Click **OK** to display the following changes on the **Users** tab:

- New users' login IDs are displayed in green.
- Users to be removed are displayed in gray, with a strikethrough.
- Modified users' login IDs are displayed in blue.

Once the Users pane refreshes, these change indicators are removed.

5 Review your changes.

We recommend using the **Show Pending Changes** checkboxes at the top of the **Users** tab to quickly review your changes. Check the **Added**, **Removed**, or **Modified** boxes to show only those users. Uncheck the boxes to show all users.

6 Revert any pending changes by right-clicking one or more rows in the table and selecting one of the following from the context menu:

- **Remove** – Remove the selected users from the user list
- **Undo Remove** – Restore the selected users marked for removal
- **Undo Edits** – Remove any changes made to a user via import or manual edits

Ctrl+click or Shift+click to select multiple rows.

7 Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes. Change indicators are removed.

If you deleted users, any aPriori licenses allocated to these users are released and available for assignment to other users.

Editing User Information

You can edit a user's information and group membership.

To edit a user

1 Select a user on the **Users** tab and click **Edit User** to display the **Edit User** window.

2 Edit the information in the fields.

For more information about each field, see Adding User on page 19.

3 Click **OK** to display the user's Login ID field in blue.

4 Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes.

Change indicators are removed.

To edit a user's group membership

1 Select a single user on the Users tab and click **Groups** to display the Groups window.

2 Check or uncheck groups as necessary to reflect the user's desired membership.

3 Click **OK**.

4 Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the Publish Changes window, then click **Yes** to publish your changes.

Note: If Access Control is implemented at your site, use the Groups tab to set permissions, and to add or delete multiple users at one time.

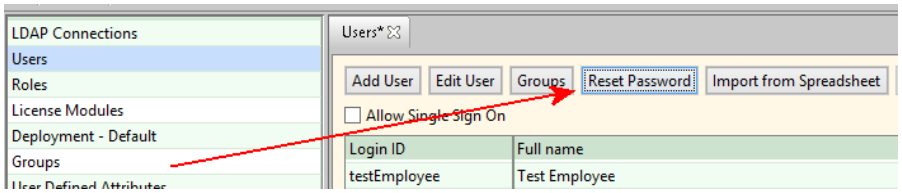
Resetting Passwords

There are different ways for a user password to be changed:

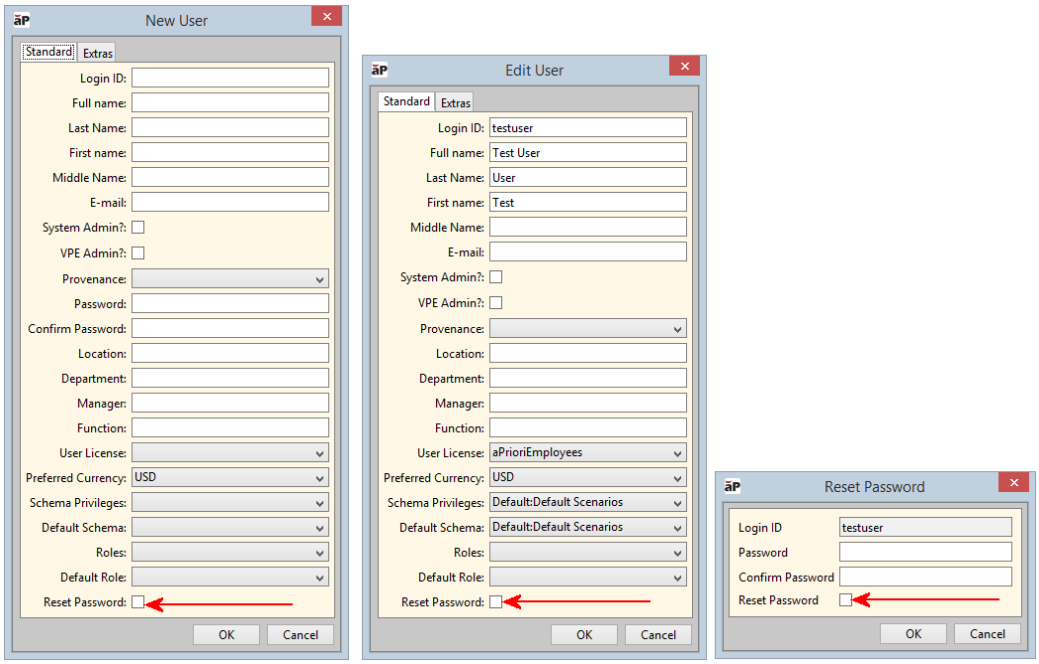
- A user can change their own password from the **Tools > Reset Password** menu item on the main UI window.
- You (as an administrator) can force a new user to change their password the first time they successfully log in with a temporary password, by checking the **Reset Password** box on the **New User** (**Add User**) dialog box.
- You can force an existing user to change their password the next time they log in with their existing password, by checking the **Reset Password** box on the **Edit User** dialog box.
- You can change a user's password directly by using the **Reset Password** button on the **Users** tool bar.

Because **Reset Password** buttons or checkboxes appear on multiple locations in the **System Administration** windows, it is important to understand how they differ from each other.

Use the **Reset Password** *button* on the toolbar to explicitly *change a user's existing password*:



Use the **Reset Password** *checkboxes* at the bottom of the following dialog boxes to *force a user to reset their own password* the next time they log in: **Add User** (**New User**), **Edit User**, **Reset Password**.



To reset a user's password

1. Select a user on the **Users** tab and click **Reset Password** to display the **Reset Password** window.
2. Enter new password and confirm it.
3. If you want this to be a temporary password that the user must change to something of their own choosing after successfully logging in, check the **Reset Password** box at the bottom of the dialog.
4. Click **OK**.
5. Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes.

Removing a User

As of Release 2018 R3 SP1, removed users are handled differently than in the past. In previous releases, when you removed a user, their information was deleted from the aPriori database and they could never again access aPriori. Now removed users can be thought of as "deactivated". They are no longer able to access aPriori, but their information still exists in the aPriori database. This means that their account can be reactivated, and that their historic activity can be accessed by Cost Insight Admin and Cost Insight Report.

Notes:

1. Along with this functionality, aPriori has introduced a new system variable to control whether removed users appear in searches:

   `includeRemovedUsersInSearchCriteria` (default value is "false")

2. When you deactivate a user, any assigned license is unassigned from that user. You reactivate the user by assigning an active license to that user (see below).

To remove a user

1. Right-click the user in the table that you want to delete and select **Remove** from the context menu.
2. The user's Login ID in the table will be displayed in gold ~~strikethrough~~. Other entries in the row will show black ~~strikethrough~~ or will be blanked or greyed. Removed users will also appear with ~~strikethrough~~ text in future Search results.
3. Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes.

   Once published, the removed user Login ID will be displayed in black ~~strikethrough.~~

4 To toggle the display of removed users, check or uncheck the box to the left of **Show removed users**.

Reactivating a Removed User

It is possible to reactivate a user that has been removed, by associating the user with an active license.

Note: If your site implements Access Control and group membership is automatic for a particular group, running an LDAP Sync automatically re-adds any reactivated users to the groups they belonged to.

If your site does NOT implement Access Control, or group membership is Manual or None, reactivated users are simply added to All Users, and must be manually re-added to the groups to which they should belong.

To activate a user

1 In the System Administration User management table, ensure that **Show Removed Users** has been checked.

2 Right-click the removed user that you want to activate. (Removed users appear with ~~strikethrough~~ login IDs and names.)

3 Click **Activate**.

4 From the drop-down menu, select the aPriori license that should be associated with this user, then click **Ok**.

5 Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to display the **Publish Changes** window, then click **Yes** to publish your changes.

Managing Single Sign On options

Automatic login to aPriori allows users to authenticate once and thereafter start aPriori without the need for re-entering username and password, for as long as the authentication is valid. As of Release 2018 R1, aPriori Professional provides two approaches to automatic logins:

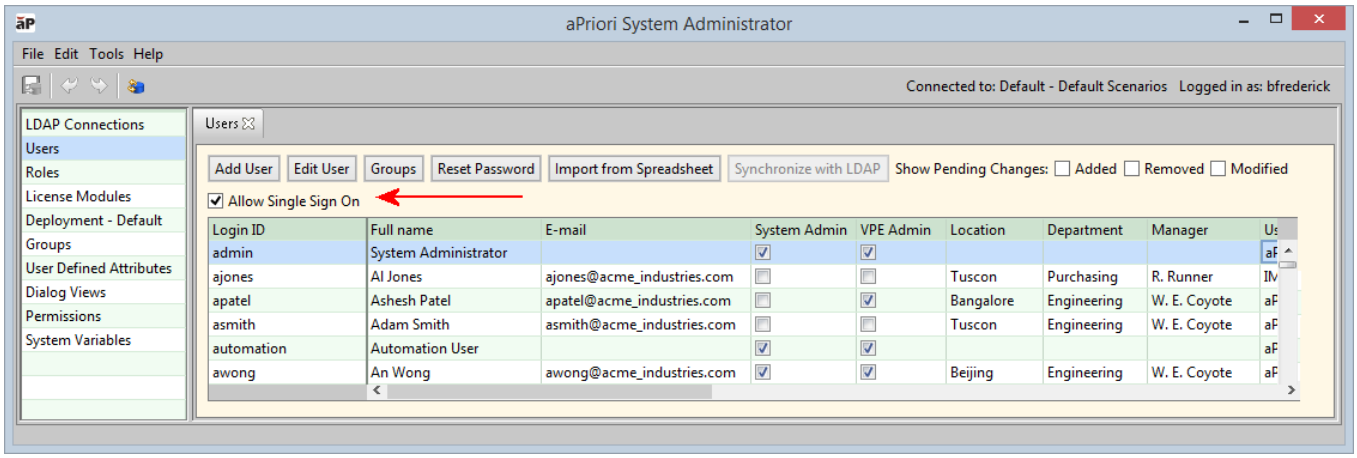
- Through LDAP authentication with Kerberos (see *LDAP Authentication* earlier in this chapter)
- By enabling the **Allow Single Sign On** checkbox at the top of the **aPriori System Administrator Users** tab (see the rest of this section)

Note: Prior to Release 2018 R1, aPriori automatically logged users in if their aPriori credentials matched their Windows credentials. This default behavior (and any other automatic log in approach that may have been implemented for your installation) has been discontinued as of 2018 R1 and should be replaced by the current approaches.

Single Sign On (SSO) allows a user to log into their computer once and then log into other applications and data sources to which they have access, without the need to re-enter their credentials each time. For example, a user can log into the Windows operating system on their laptop, and then start aPriori without an additional login.

aPriori SSO requires that your environment uses Active Directory with Kerberos authentication caching enabled (typical for a modern Windows environment).

To enable or disable SSO, use the **Allow Single Sign On** checkbox at the top of the **Users** tab. This setting is enabled (checked) by default.



Once SSO is enabled, the exact behavior your users will see when starting aPriori depends on how they log into your company's authentication environment. Here are three common environments:

- **Domain Login –** The user logs into their laptop (working from the office, or at home and logged into the company's VPN). In this case their credentials are validated using Windows Authentication (Kerberos + Active Directory). When they start aPriori it uses their cached credentials which are retrieved from the Windows Operating System. aPriori uses this ID to find a valid cached Kerberos ticket and then it validates that the user is a licensed user in the aPriori user

database. This works if the user's aPriori login id matches their Windows username.

- **Local Login –** The user logs into their laptop (working at home but not logged into the VPN). In this case their credentials are authenticated by the local system using the SAM (Security Accounts Manager). When they try to start aPriori, it will not find a valid Kerberos ticket. The aPriori log in screen appears and the user is required to enter their username and password.

  Note: This description is a little over-simplified. There are situations where a user can log in successfully as an SSO user to aP Professional even when disconnected from the VPN and fully disconnected from the network. If they had connected earlier and obtained a valid Kerberos ticket (and their DB is local to their install), they may be able to login in again without connecting through VPN if the ticket has not yet expired (typically 10 hours).

- **Remote Login –** The user uses Remote Desktop Connection to log in to another computer or VM within the company's firewall. In this case the remote system is on the internal network and the credentials the user enters when logging in remotely will be authenticated through the domain controller. This behaves the same as Domain Login above: when the user starts aPriori, they do not need to re-enter their credentials.

  Note: Remote Desktop users should *sign off* rather than *disconnect* from the Remote Desktop Connection. A disconnect can result in zero cached tickets for the next login to the Remote Desktop. – Causing SSO to behave unexpectedly for the user.

Note: You should educate your users that when they work remotely, they must be logged into the company VPN or else they *will* typically need to enter their aPriori credentials. This can be confusing to users who only rarely work remotely, since they typically do not need to enter aPriori credentials and may not even remember them when they are unexpectedly prompted for them.

Single Sign On debugging

To assist tracking down problems when configuring Single Sign On, you can increase the level of SSO-related messages that are recorded in the logs by uncommenting the following line:

```
#log4j.logger.com.fbc.datamodel.auth.kerberos=debug
```

in the following file:

```
<aP_install_dir>\install\install-files\log4j.properties
```

aPriori strongly recommends that you do so with the assistance of aPriori Customer Support.

Single Sign On and aPriori Command Line Utilities

Single Sign On also applies to the aPriori following command line utilities:

- Bulk Costing (bulkLoad.cmd)
- BOM Loading (bomLoad.cmd)
- Client Compatibility Updater (updateCompatibleVersions.cmd)
- Doc Cache Cleaner (cleanDocCache.cmd)
- Doc Cache Updater (updateDocCache.cmd)
- Watch Point reports (runAprioriReport.cmd)

In general, when Single Sign On is enabled, these scripts will attempt to use the signed-in Windows user credentials, unless you override this behavior by providing the credentials on the command line or in a relevant properties file.

The order of precedence for using credentials is:

1. Command line arguments.
2. Stored in the relevant properties file (if applicable).
3. Signed-in Windows credentials from SSO.

Note that if you are upgrading to a release that uses SSO and you have existing command line scripts, you will need to review them and determine if:

- You want or need to remove current credentials to make use of SSO.
- You need to reorder the position of existing credentials on the command line as order may have changed as part of the implementation of SSO.

See the *aPriori Professional Installation Guide* for upgrade details.

Managing license modules

The aPriori System Administrator allows you to control the licenses required to use aPriori. The aPriori license information includes user licenses and module licenses. Licensed modules include:

- Regional Data Libraries
- Manufacturing Cost Models, such as Sheet Metal I (soft-tooled processes), Sheet Metal II (hard-tooled processes), Plastics Molding, Surface Treatment, etc.
- Functional modules, such as Bulking Costing & Analysis, Catalog Part Loader, VPE Creator, and Cost Model Workbench

When a user logs into aPriori, the software validates the license in two ways:

- At login, aPriori checks that the user is associated with a valid user license.
- When costing, the application checks that the user is licensed for that particular module.

A deployment can have multiple license files. For example, if you purchase a subscription and then extend the number of licenses in your deployment, aPriori will provide a second license file with the additional user licenses.

To manage licenses

Click **License Modules** in the Navigation pane to display the **License Modules** tab.

The name of the company that owns the licenses and the following user license information is displayed in the **User Licenses** list.

Column	Description
License Name	Descriptive identifier
Num Users	Maximum number of users that can be assigned
Num Purchased Users	Number of users, including administrators, purchased for the license
Num Users in Use	Number of aPriori users with assigned licenses
Expiration Date	Date and time the license expires
License ID	Unique identifier for each type of license generated at the time of purchase
Signature Data	Digital signature for the license

The **License Modules** list displays the licensed modules associated with the license selected in the **User Licenses** list.

Users can only calculate costs for the process groups for which they are licensed. For example, a **Sheet Metal** license module is required to cost any sheet metal component. If a license is nearing its expiration date, or if you need additional user licenses, please contact your aPriori Account Executive or aPriori Support.

To load a new license

1. Select **File > Load License File** from the System Administrator menu bar to display the **Select a license file to load** window.
2. Navigate to the folder containing your license file, select the file, and click **Open**.

   The license file is added to your environment and is displayed in the **User Licenses** list.
3. Assign users to the license.
4. Select **File > Publish Changes** from the System Administrator menu bar or click  in the toolbar to save your changes.

Managing deployments

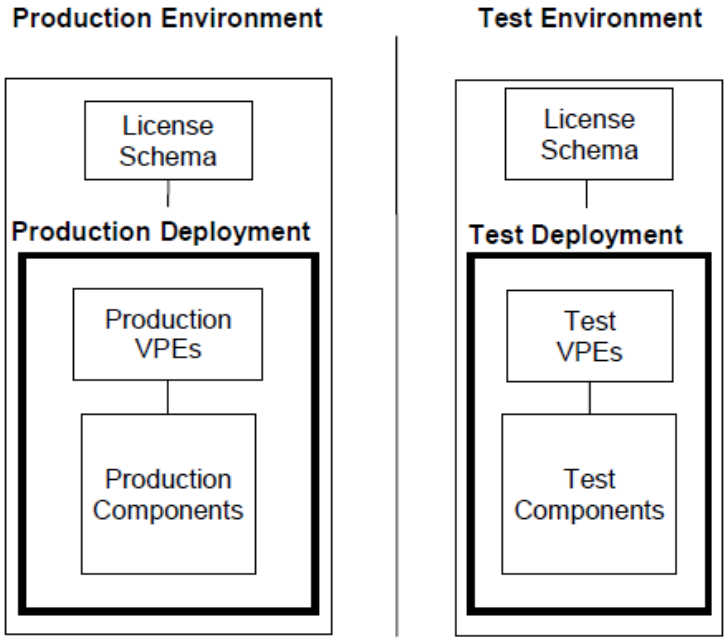
aPriori provides flexible configuration options to best meet your security and operational requirements. Separation of data is achieved by creating a number of *deployments* within aPriori. Each deployment contains costing algorithms and previously costed scenarios. Access can be restricted so that users can only access certain deployments. This functionality provides a solution for companies that have groups which do not want to share information or must segregate products and components due to regulations such as iTar (International Traffic in Arms Regulations).

Here are some examples of how this can be used:

- A company can segregate users and the parts they cost, while maintaining a single repository for VPEs that can be accessed by all users.
- Test and training environments can be kept entirely separate from the production environment, allowing users to work without any risk to production data.
- Users who are distributed globally and who do not have the network performance to connect to a central database, can maintain a local deployment for each team.

Recommended Deployment Architecture

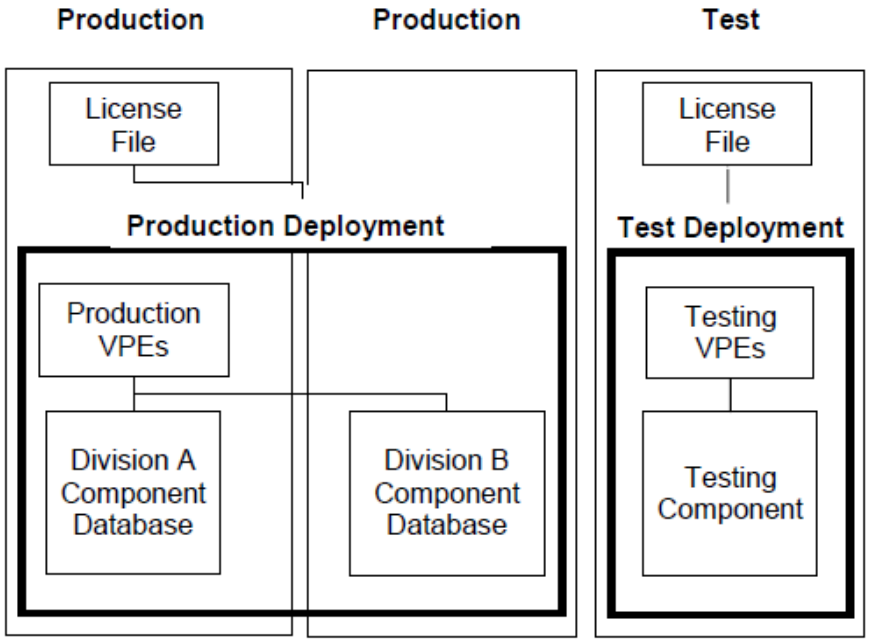
The aPriori *Installation Guide* provides an example of a very basic recommended deployment, with separate environments for production and testing.



Production Environment– The environment where users' day-to-day work is completed and saved.

Test Environment – The environment where VPEs, UDAs, etc. are edited and tested. The recommended workflow is to export VPEs from the Production Deployment, import into the Test Deployment, edit and test in the Test Deployment and then Import into Production Deployment.

This chapter provides discussion and examples of more complex deployments. For example, the following deployment shows how to segregate data between different divisions of the same company:



There are several use cases for an architecture such as this:

- A company that has several competitive divisions, or areas of business that must be separate for adherence to regulation, can configure a production deployment with multiple component databases**.** In this type of deployment, users are granted permission to a specific component database. If you have super users or managers who should be able to access all component databases, you can assign permission to access multiple component databases.
- A company that has geographically remote users may find that some users cannot achieve appropriate network performance when connecting to a single server. In this case, the company can use deployments to distribute VPE and component schemas to servers that are closer to those end users. This could be used in conjunction with aPriori's separately-licensed Scenario Synchronization module for keeping component schemas synchronized if that is a requirement.
- A company that develops VPEs internally should develop their VPEs in a separate “development” deployment. Once these have been verified, the VPEs can be published to their end-user deployment. This can be linked the same license file as the end-user production deployment.

  If this “development” deployment is linked to the same license file as the end-user production deployment, then it should not be used to make changes to Access Control Objects, UDAs, etc. These objects are shared across all deployments which share a license schema, and changes could disrupt a deployment being used for costing in production. These objects should only be developed in a separate deployment with a distinct license file.

Please contact aPriori support if you have questions about deployments, or any other aspect of aPriori IT architecture.

What is Stored in the aPriori Database?

To understand how data stored in the aPriori database can be segregated, you must first understand how that data is stored.

There are three types of data stored in the aPriori operational database:

- **License Data –** The license data for aPriori resides on the database. Each aPriori license specifies the number of named users that are allowed to log into the database, and aPriori licensing is named-user licensing. When a user attempts to log into the aPriori client, the user details that are provided are validated against the list of valid users that is stored in the database. If the user details match a user in the database, then that user is logged in.

  As the aPriori license resides on the database, this means that a user can log in from any PC that the aPriori client is installed on.
- **Virtual Production Environment (VPE) data –** A VPE contains the algorithms and rates that are used to calculate the cost of a scenario. As the VPE is stored in the database, this means that all users can always use the most up to date algorithms and data
- **Component data –** Whenever a scenario is costed in aPriori, the results are saved to the database. This allows different users to share results and perform analysis on a wide range of scenarios from around their company.

When aPriori is first installed, the three types of data reside within the same database schema, which is the schema specified in the local file *hibernate.properties*. This is the *Master Database Connection.*

aPriori Licensing

When the aPriori client first starts, it always initially connects to the schema specified in the local file *hibernate.properties*. The license stored within this schema is the license that the user is verified against. Even when multiple deployments are being used, or when multiple databases are being used globally, license management can therefore still be maintained centrally on a single database.

Note: You should never configure Test and Production deployments on the same installation, sharing the same license file. Although this deployment strategy was once supported, you should always place Test and Production deployments on separate installations. You can use multiple instances of the same license file (assuming that your database license agreement allows for this) on different installations, just do not share the same instance of a license file between Test and Production deployments on the same installation. This limitation does not prevent you from sharing the same license schema across multiple production deployments as discussed later in this chapter.

Once the user has successfully logged into this server, the user may then be redirected to other deployments. Deployments can be on the same server, or they can be on different servers.

Remote User Teams

Users in global companies may find that they do not have sufficient network performance to work on some of the servers within the company. By redirecting users to different schemas, users can also be redirected to servers that are closer to them geographically. User licensing can still be maintained from one server and one license file, as the license check does not require a particularly fast network connection.

If the users in different regions share cost data, and they need to regularly import and export scenarios to stay current with their colleagues, then different deployments can be kept consistent using the aPriori Scenario Synchronization functionality. See the *Guide to Scenario Synchronization* which is available from the aPriori Community Site (https://www.apriori.com/customer-knowledge-database/product-documentation/) or contact your aPriori representative for more information.

Granting Permissions to Deployments and Access Control

One of the reasons to use deployments is to control access to different schemas. Consider the following example of two schemas used by an aerospace company:

- One schema is called “Civilian” and contains data pertaining to parts used in civilian planes.
- The other schema is called “Military” and contains data pertaining to parts used in military planes.

It is possible that data in the “Military” schema can only be accessed by users with the appropriate security clearance, or users who are resident in a certain country. In this case, access to the “Civilian” schema could be granted to all users, but access to the “Military” schema would only be granted to the appropriate users.

Using deployments to segregate data performs a similar role to aPriori *Access Control*, as it allows administrators to specify what data can be accessed by each user. Using deployments is limited however, as access can only be controlled at the schema level, and any user granted access to a schema will have full access to that schema.

Access Control allows administrators to grant permissions on a much more detailed level. Users and groups can be granted a range of permissions such as “create”, “delete”, and “cost using”, and these permissions can be granted on a much more detailed level. See the Access Control chapter for more details.

Managing Deployments as a System Administrator

When a new deployment is created, a system administrator specifies the *VPE Schema* and the *Component Schemas* for the deployment. These are database schemas in MySQL or Oracle and are defined using the same fields that are used during installation. These schemas can use the same connection as the *Master Database Connection*, or can be entirely different schemas, perhaps stored on a different database.

When a user is accessing a deployment, that user will access License data, VPE data, and Component data, but this data may reside in different locations.

- **License Data –** The user login details are always verified using the master database connection specified in *hibernate.properties.*

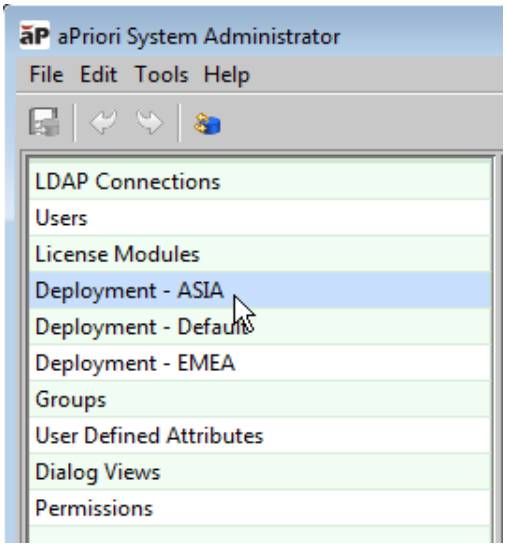
- **VPE Data –** The VPEs that a user can access are stored in the schema defined by the VPE Schema in the deployment that they are using. If this is not otherwise specified, then the master database connection is used.
- **Component Data –** The component data that a user can access are stored in the schema defined by the Component Schema in the deployment that they are using. If this is not otherwise specified, then the master database connection is used.

The deployments that an individual user can access are controlled by the aPriori System Administrator. Deployments can therefore be created to contain information which is specific to a particular group of users, and access to that deployment can be restricted so that only those users can access that deployment.

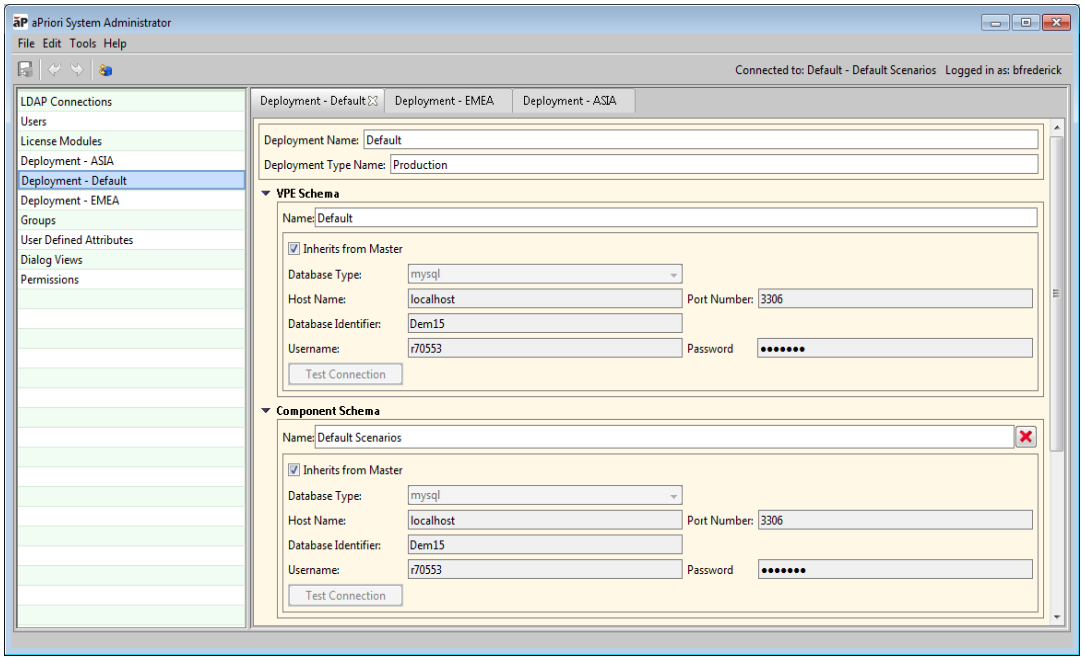
To view the deployments

1 From the aPriori client, click **Tools -> System Admin Toolset**.

2 When the **aPriori System Admin Toolset** window opens, click the **System Administrator** button. In the resulting **aPriori System Administrator** window, the deployments you have are listed in the left column.



3 Select a deployment to display its detailed properties.



To create a new deployment

1 From the aPriori client, click **Tools -> System Admin Toolset**.

2 When the **aPriori System Admin Toolset** window opens, click the **System Administrator** button.

3 In the resulting **aPriori System Administrator** window click **File -> Create Deployment**

aPriori creates and publishes the new schema with default values and displays the details in the right panel.

4 Edit the properties as necessary and click the **Publish Changes** button () when done. See the following section (Deployment settings that can be edited) for more information.

Deployment settings that can be edited

The top two fields contain general deployment values that can be edited:

- **Deployment Name –** The name of the deployment that you are creating
- **Deployment Type Name –** This is descriptive of the deployment and is for information purposes only.

Each deployment contains one VPE schema, and one or more component schemas.

- The VPE schema specifies the database schema from which VPE data will be available to a user.
- The component schemas specify what database schemas the user will be able to read component data from. Users are granted access to an entire deployment, so they therefore may be granted access to a number of component schemas.

Each schema contains the following settings:

- **Schema name –** This is the name of the schema and is used for display purposes when managing what deployments users can access.
- **Inherits from master –** If this option is set for a schema, then the database connection data is copied from the local file *hibernate.properties* and cannot be edited further. This is set on a schema-by-schema basis.
- If the option **Inherits from master** is not set, then the following settings can be edited for a schema. These are the same settings that are used to specify the database when installing aPriori, and they specify the database schema that will be used.
    - **Database type –** MySQL or Oracle (If the database is Oracle, then there will also be an option to select whether the connection is specified by SID or Service Name.)
    - **Host name**
    - **Port number**
    - **Database identifier**
    - **Username**
    - **Password**

If the **Inherits from Master** option is not checked, then a **Test Connection** button becomes enabled. This allows you to test if the connection details that have been entered result in a successful connection.

To add or remove component schemas

1 To add a new Component Schema, click the green "**+**" symbol at the bottom left of the screen.

2 To remove a Component Schema from the deployment, click the red "**X**" symbol at the right of the Component Schema name. (This button is only enabled if multiple Component Schemas exist in the deployment.

Editing User Settings

Once you have created deployments, you must assign rights to users to access these deployments. This is done through the **Users** tab in the **System Admin Toolset** window. (See *Managing users* for general information about this tab.)

To edit user settings for deployments

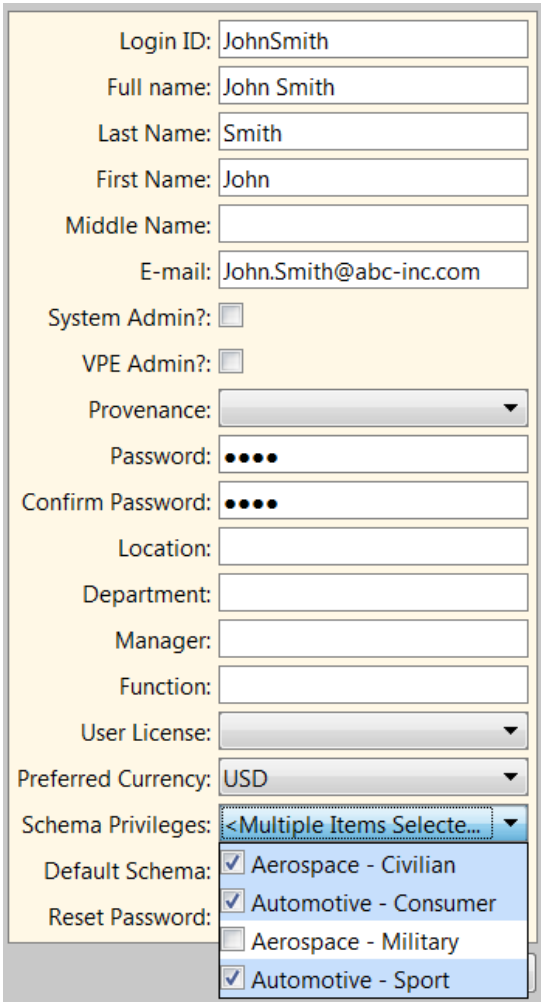
1 From the aPriori client, click **Tools -> System Admin Toolset**.

2 When the **aPriori System Admin Toolset** window opens, click the **System Administrator** button.

In the resulting **aPriori System Administrator** window click **Users** in the left column.

3 Select the user in the right pane. (You can select multiple users by holding down the **Ctrl** key as you click usernames.)

4 Click the **Edit User** button to display the **Edit User** dialog box.



5 Note the following drop-down menus toward the bottom of the dialog:

- **Schema Privileges**
- **Default Schema**

These fields are explained in the following section.

Using the Schema Privileges and Default Schema fields

The **Schema Privileges** drop-down can contain a number of items with check boxes next to them. Each of these represent the name of a deployment, followed by the name of a component schema in that deployment. By checking the box for each item, you grant that user the ability to cost using the VPEs associated with that schema, and to view the component schema within the deployment.

This means that users can be restricted so that they can cost using the VPEs associated with a deployment, without being able to view all the component data.

Consider the example shown in the preceding image:

- There are two deployments: **Aerospace** and **Automotive**
- The **Aerospace** deployment contains two component schemas: **Military** and **Civilian**
- The **Automotive** deployment contains two component schemas: **Consumer** and **Sport**

This user has been granted access to all data *except* the component data in the **Aerospace – Military** schema.

The **Default Schema** drop-down specifies what the user will access when first logging into aPriori. This contains a list of all the items that were selected for that user in the **Schema Privileges** drop-down. The system administrator can select what data users will see when they first log into aPriori.

A user may change what schema they are connected to at any time, by clicking **File->Switch component schema…** from the aPriori client toolbar and selecting from a list of all the schemas to which the user has access.

Managing groups

Note: *As of aPriori 2015 R1, the Groups tab has become a major tool for configuring Access Control. See the Groups Tab section of Chapter Access Control for more information about groups.*

Managing User Defined Attributes (UDAs)

When you create a component within aPriori, a standard set of attributes are provided by default. These attributes are generally descriptive and are used for aPriori's search and report features. On the **User Defined Attributes** tab, you can create additional custom user attributes ("UDA"s) to support your company's specific design and cost management practices.

Once created, User Defined Attributes can become a central part of user workflow and be found in many areas of the application, including:

- **Search** – Users can search for components based on UDAs.
- **Administrative Information** – Users can enter values for UDAs on the Administrative Info tab of the Cost Object Info window or under sections such as "Company Defined Attributes" of the Cost Guide. Users must enter values for required attributes to cost a component.
- **Part Details/Assembly Details** – UDAs can be displayed on new views on the **Part Details** and **Assembly Details** tabs.
- **BOM Loader** – Users can map UDAs when loading components via a bill of materials. Users must map and populate required attributes to load components.
- **Bulk Costing** – Users can enter values for UDAs when bulk costing and must provide values for required attributes. Users that want to bulk cost existing components and use existing required attribute values must do one of the following:
  - Collect all scenarios into a container (assembly, roll-up, or dynamic roll-up) and use Deep Costing. The existing UDA values will be pulled in for all scenarios.
  - If all scenarios have the same scenario name, ensure that that scenario is used when creating the new bulk cost group. All existing UDA values will be pulled into the bulk costing UI.

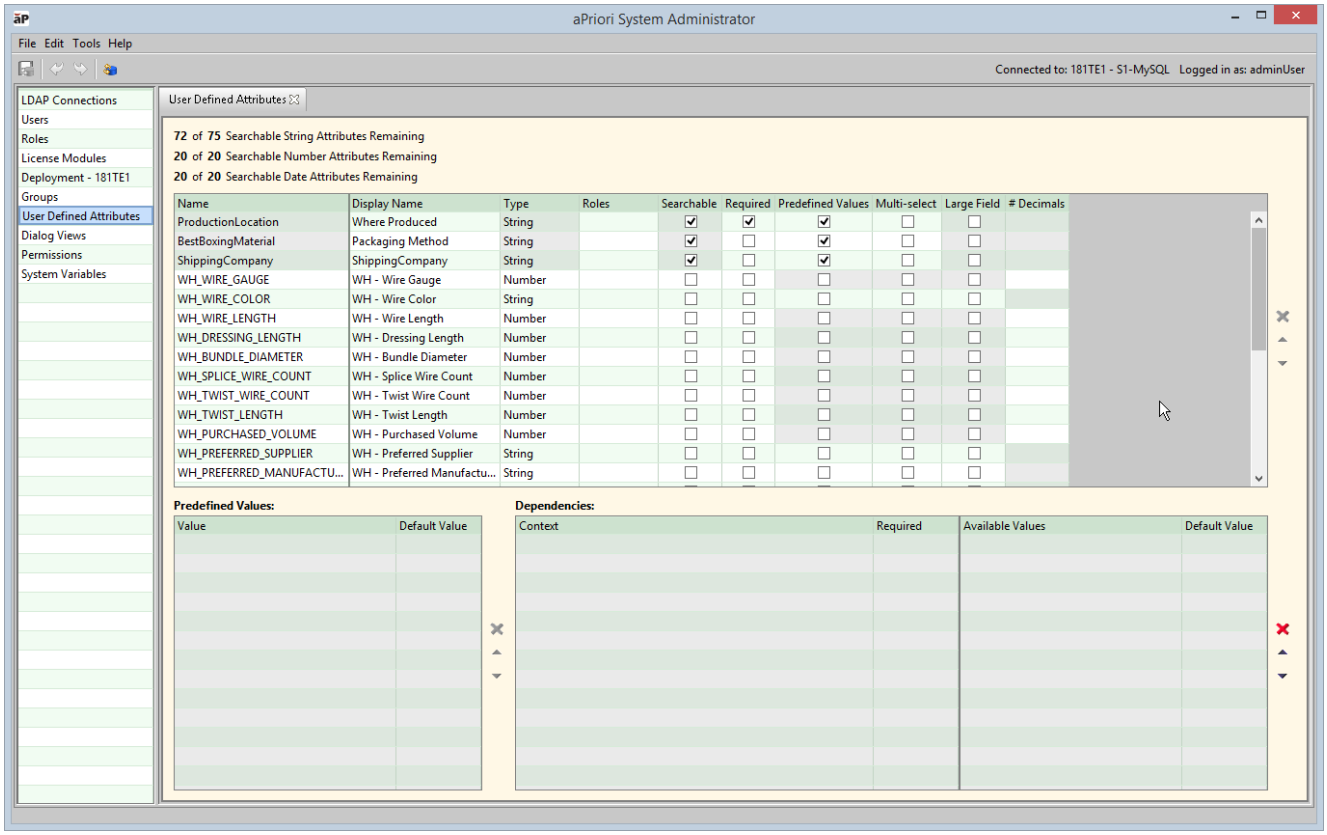
User Defined Attributes can also perform a number of other useful functions:

- CAD properties can be mapped directly to UDAs (see Using CAD Properties).
- UDAs can be used to allow scenario-specific site variable overrides. See *Using UDAs for Scenario-Specific Site Variable Overrides* below, and the "Process Group Site Variables" chapter of the *aPriori Professional VPE Administration Guide* for more information.
- User defined attributes can be evaluated by the aPriori proprietary scripting language that generates cost estimates. This means that attributes can be referenced by cost models and can influence cost estimates. Cost models in your VPEs can reference the values of user defined attributes by appending the Name of the attribute (not the Display Name) to `part.customAttributes`.
- As of Release 2018 R1, UDAs that have pre-defined values can make use of *Dependency Contexts* so that System Administrators can define behavior based on selected inputs and other attributes. This is a powerful feature for guiding your users through the selection of critical values and preventing them from selecting

unavailable or nonsensical options. This capability is described in *Understanding Dependent UDAs* below.

To manage user defined attributes

Assuming that the aPriori System Administrator window is already displayed, click **User Defined Attributes** in the Navigation pane to display the **User Defined Attributes** tab. The screenshot below shows several UDAs. Your display will be blank if you have not yet defined any attributes.



To create a new user attribute

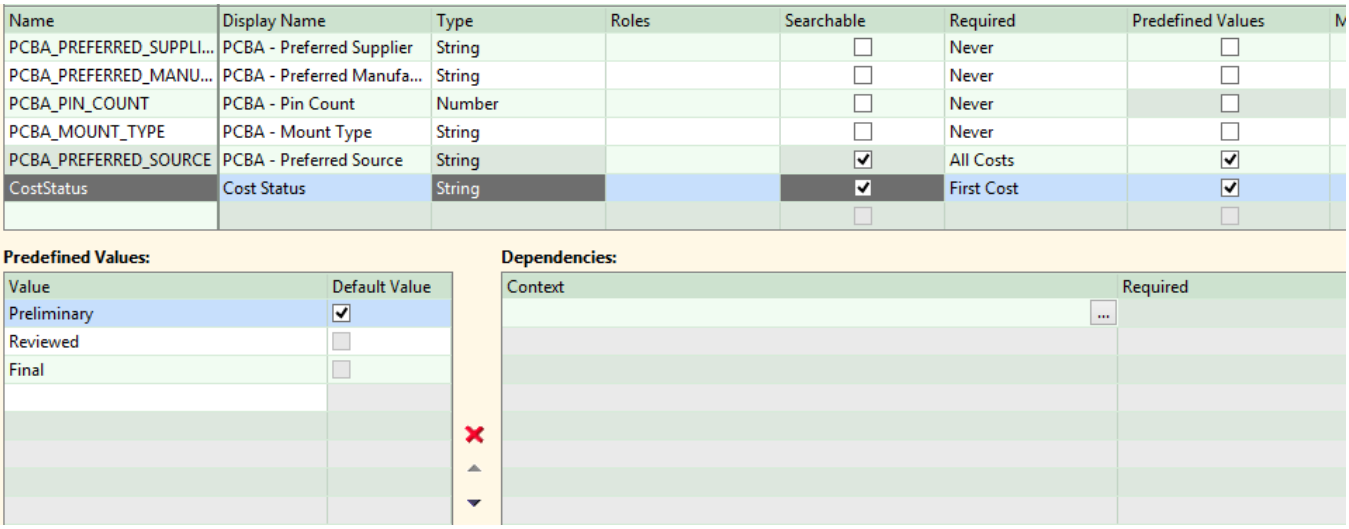
1 Enter information on a blank line in the table in the upper half of the tab. (If you already have several attributes defined, you might need to scroll to the bottom of the table to access a blank line.) Fill in the fields under the columns as described in the table below.

Column	Description
Name	Required. Enter the name of the attribute as you want it to be referenced in aPriori's proprietary scripting language. Characters are limited to numbers, letters, and underscores.
Display Name	Enter the name of the attribute as you want it to appear within the aPriori interface, such as on the Search and Cost Object Info windows.

Type	Required. Double-click in the field and select one of the following types from the drop-down list: ▪ **String** – Allows the user to enter text into a field. Users can enter text directly or select from predefined values. ▪ **Number** – Allows the user to enter a number into a field. You must select this type if you want aPriori to use the value in this field for cost calculations. ▪ **Date** – Allows the user to enter a date with a calendar. ▪ **User List** – Allows the user to select one or more users from the aPriori user list.
Roles	For Cost Insight Design and future enhancements. As of Release 18.1, the only option is **Design** (for "Design Engineer"). Do **not** use this field unless you have been directed to by aPriori personnel or documentation.
Searchable	Check this box to make the attribute searchable within aPriori. The number of searchable attributes supported within aPriori is limited for performance reasons. Only 75 string attributes, 20 number attributes, and 20 dates can be searchable. (A searchable user list attribute is counted as a string.) This information is displayed at the top of the User Defined Attributes tab. Note: Once a UDA is marked searchable, it can't be deleted or made unsearchable
Required	Select an option from this drop-down menu to determine if and when this attribute MUST be set when costing. The options are: ▪ **Never** – the default ▪ **First Cost** – the attribute must be set only the first time a component is costed ▪ **All Costs** – the attribute must be set whenever a component is costed. Note that First Cost is the behavior found in releases prior to 2018 R1. If you upgrade from a previous version, your existing required UDAs are migrated to this option. Required attributes display a red asterisk (*) in the **Cost Guide** and **Cost Object Info** windows. aPriori recommends that you use consistent requirement behavior across UDAs to prevent users from being confused about varying behavior. **Note:** Behavior specified for Dependency Contexts takes precedence over the Required behavior for the UDA, unless all the dependencies evaluate to false, then the UDA Required setting is used.
Predefined Values	Check this box to allow your users to select from a drop-down list. Available only for attributes of type String and User List. See the step following this table for configuring predefined values.

	Note: As of Release 18.1, Predefined values can also be configured with *Dependency Contexts* to help you guide your users to select logical and correct values when costing. See *Understanding Dependent UDAs* for details.
Multi-select	Check this box to allow the user to select multiple predefined values. Available only for attributes of type User List and String (when you check the Predefined Values box).
Large Field	Check this box to allow the user to enter unlimited descriptive text into a field. Available only for attributes of type String when you uncheck the Predefined Values box.
# Decimals	Enter the number of decimals you want to display in aPriori. For example, if you enter 2 as the number of decimals for the attribute, and the user enters 1 in the attribute's field, aPriori will display 1.00.

2 If you are defining **Predefined Values**, use the **Predefined Values** panel to enter the values you want to appear in the drop-down list. In the following screenshot, we have defined "CostStatus" to be searchable and required and given it three predefined values.



You can optionally check the **Default Value** box for a value you want to pre-select in the drop-down list. You must uncheck the selected default value to select a new default value. Use the Up and Down arrows to reorder the members of the list, and the red "x" to remove members from the list.

3 If you are NOT using **Predefined Values** with Dependency Contexts, you can stop here and select **File > Publish Changes** from the **System Administrator** menu bar or click  in the tool bar to save your changes.

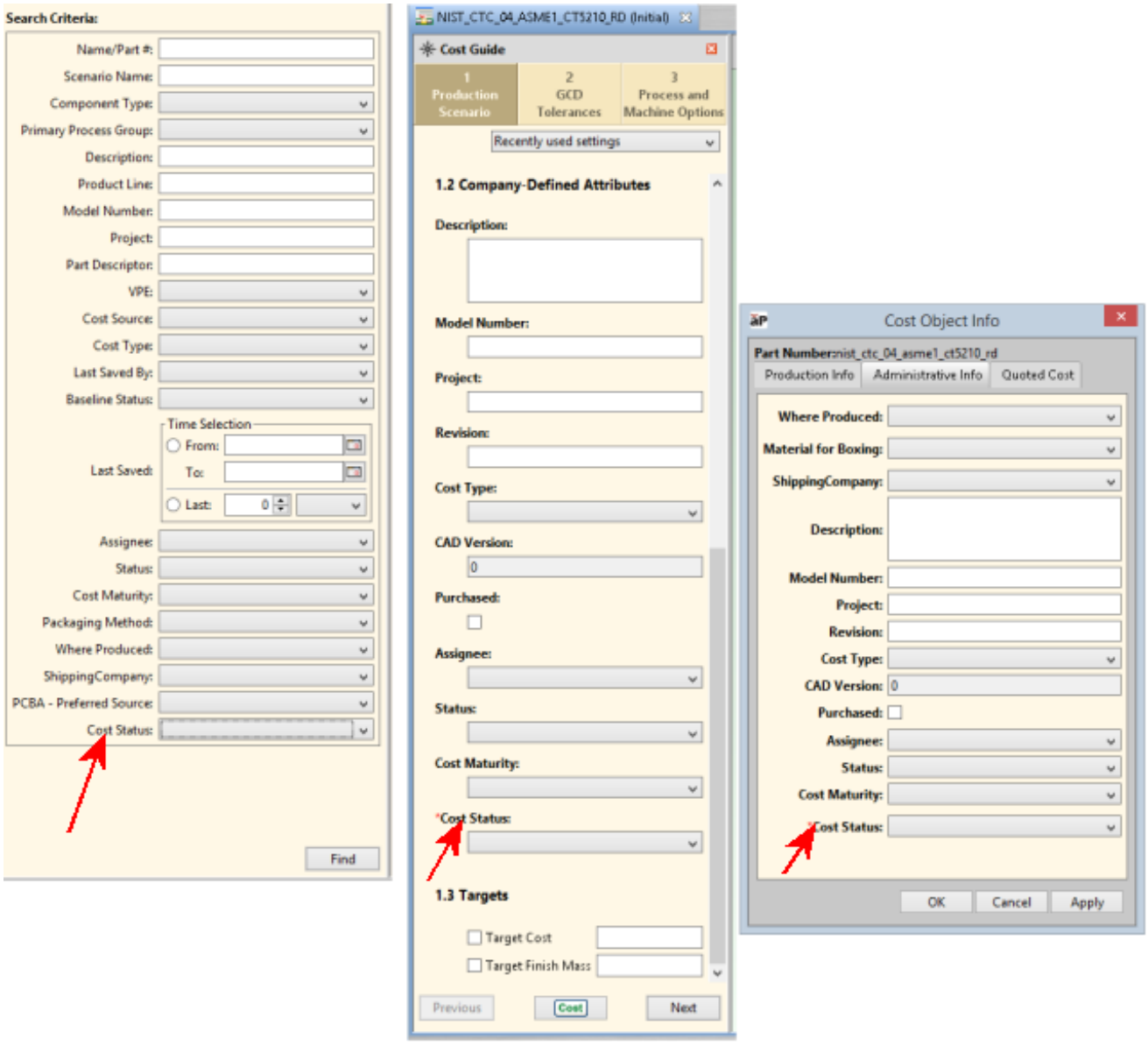
4 If you ARE defining **Predefined Values** and wish to further configure them with Dependency Contexts, proceed to *Understanding Dependent UDAs* for details.

By default, a UDA is considered "Available" but not "Displayed" when first created. So, once your UDA is created, use **Dialog Views** to determine how and where searchable and/or required UDAs are displayed on the **Search**, **Cost Guide**, and **Cost Model Info** windows. See *Managing dialog views* for details.

Note: If you have defined a UDA to be “Required” but do not make it “Displayed” with **Dialog Views**, your users will receive an error message whenever they try to cost a component, saying that a UDA that they cannot see is required, and advising them to contact you.

How UDAs Appear in the User Interface

Assuming that you make your UDA “Displayed” in both the Search Window and the Cost Guide & Cost Model Info window (but do not make any other changes in **Dialog Views)**, here are some of the default locations where your new UDA will appear. In this example, the new attribute is a basic (non-Dependent) Defined List named “Cost Status” that is both Searchable and Required. It appears (from left to right) on the Search, Cost Guide, and Cost Object Info windows.



UDA Recommended Practice

When you define a UDA that only makes sense for certain Process Groups, use Dependency Contexts (see below) so that your users can edit the UDA only when a relevant Process Group is selected. This approach isn't always possible, for example for some site variables that require fine control. But it is useful for values such as Booleans like **useShortestStockDimensionForPrimarySetup** in **Stock Machining**.

Using UDAs for Scenario-Specific Site Variable Overrides

If you define a UDA to have exactly the same name as a site variable, its value will override the site variable. This allows you to change site variable settings on a per-part basis. (The UDA does not change the site variable setting; it overrides it for this part.)

If your company has separate aPriori System Administration and VPE Administration roles, you will typically collaborate with a VPE Administrator for this process:

1. Determine which site variable should provide override capabilities. You can view site variables clicking:

   Tools > VPE Toolset >Process Group Site Variables

   and then selecting the desired Process Group from the left pane.

2. Ensure that you and the VPE Administrator understand exactly which Process Group(s) this override should be used for.

3. Using the procedure described in *Managing User Defined Attributes (UDAs)*, create a new UDA with EXACTLY the same name as the site variable. In the **Display Name** field, ensure that the label includes the name of the Process Group(s) that this override applies to.

4. Use **Dialog Views** to ensure that the new UDA is visible on the Cost Guide.

Exporting and Importing Custom Attributes

You can export custom attributes from and import custom attributes into deployments. This is useful for customers who wish to define custom attributes in a test environment and then port them into a production environment when they have been finalized.

To export custom attributes from a deployment

1. Select **File > Export > User Defined Attribute Data** from the aPriori System Administrator menu bar to display the **Export User Defined Attribute Data** dialog.

2. Specify a filename and folder for the exported attributes and click **Save**.

To import site variables into a deployment

1. Select **File > Import > User Defined Attribute Data** from the aPriori System Administrator menu bar to display the **Import User Defined Attribute Data** dialog.

2. Specify the `.uda.zip` file to import and click **Open**.

Understanding Dependent UDAs

Release 2018 R1 introduces the ability to add logic to User Defined Attributes (UDAs) that have Predefined Value lists, allowing you to define *Dependency Contexts* that determine what options are made available to your users under different circumstances.

This allows you to help narrow down the choices that your users need to make so that they do not waste time selecting values from a huge range of possibilities and help them to avoid making incorrect or nonsensical choices.

For example, say that your company needs to specify the temper applied to a given material. There might be hundreds of temper values available to choose from. By defining a Predefined List UDA to automatically narrow the available tempers based on the material that is selected, you can automate what would otherwise be a near-impossible task.

Another example could be the organizational hierarchy within a large global company that has multiple regions, with each region having multiple product lines, which in turn have multiple projects. Having a single list of all company projects could quickly become unnavigable for users looking for a particular one. But if you define lists to display only those projects that exist within a particular product line within a particular region, the choice is narrowed down to a manageable selection.

Dependent UDA Example

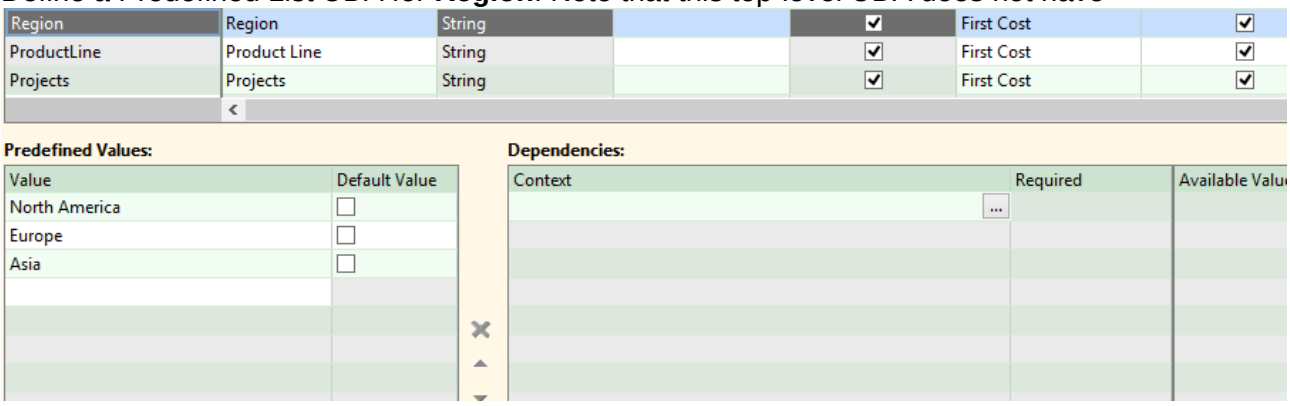
Here's a simple example. Assume that your company has three regions, each of which has a specific product line, and each product line has three active projects:

Region:	North America	Europe	Asia
Product Line	Widgets	Gizmos	Doodads
Projects	X1A X1B X1C	X2A X2B X2C	X3A X3B X3C

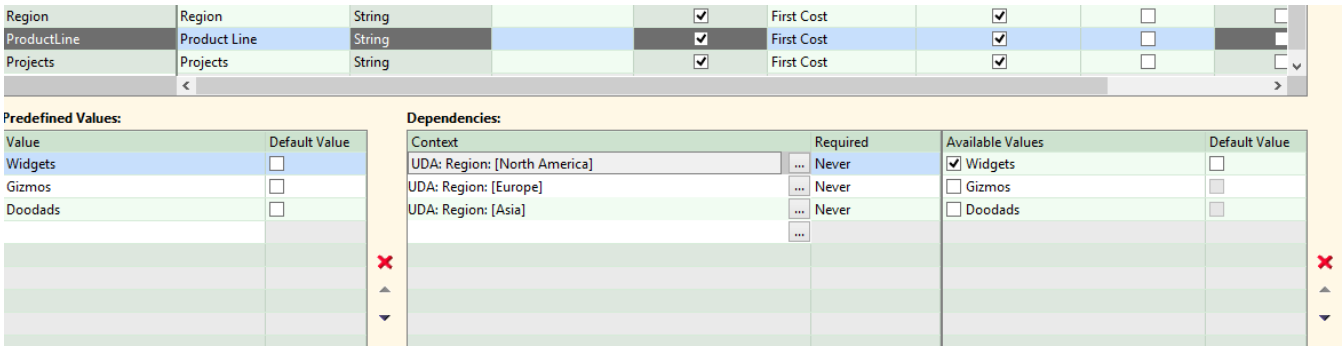
You can define three UDAs for Region., Product Line, and Projects, and give them predefined lists for the values. But if you place these lists on the **Cost Guide** and expect your users to select the correct combination of values when costing components, there's a very good chance they could get it wrong and spend a lot of time in the process. For example, a user could select "North America" for the region, and "Gizmos" for the product line, and "X3B" for the project – none of which match the other values.

Here's how you could define these UDAs with Dependency Contexts to ensure only the correct values are made available, based on other selections.

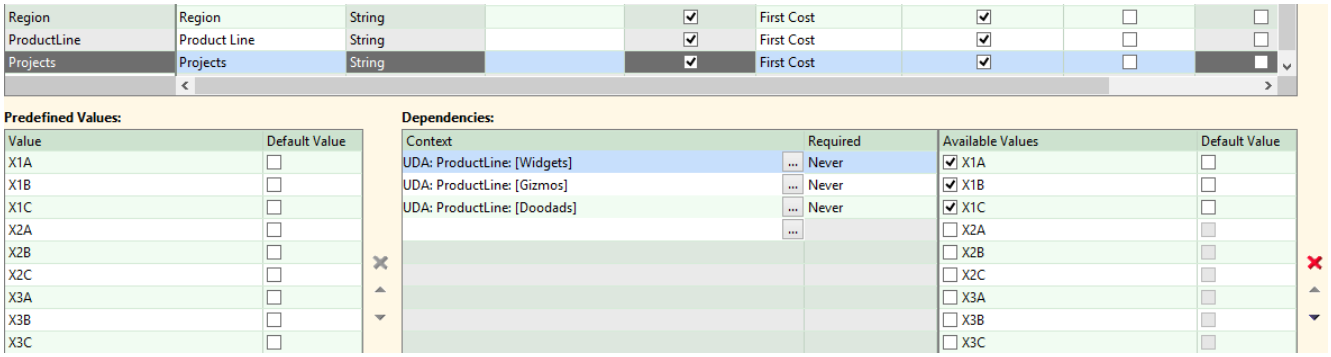
1 Define a Predefined List UDA for **Region**. Note that this top-level UDA does not have



2 Define a Predefined List UDA for **Product Line**. Note that this UDA has Dependency Contexts on **Region**. North America is shown, but you would need to define dependencies for Europe (Gizmos) and Asia (Doodads) also.

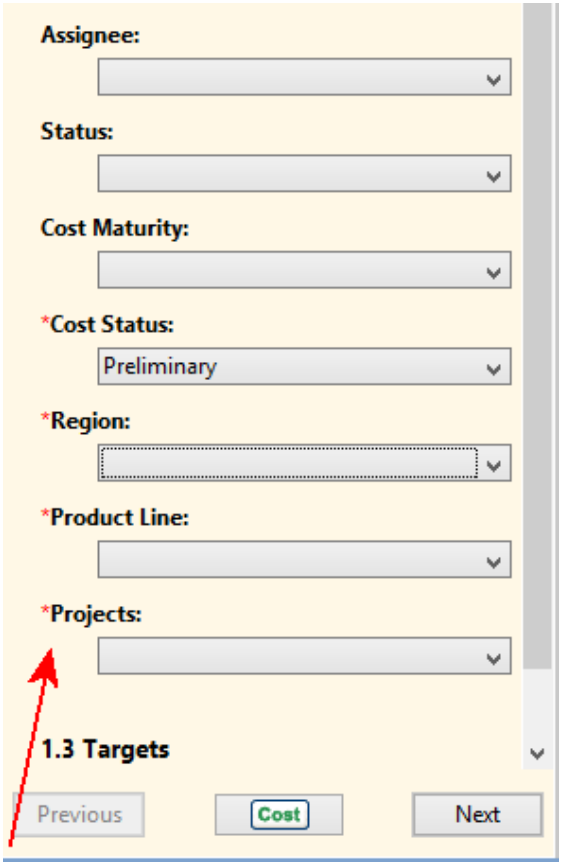


3 Finally, define a Predefined List UDA for **Projects**, which has dependencies on **Product Line**. Widgets is shown, with the three projects that belong to that line, but you would need to also define the dependencies for Gizmos (X2A-X2C) and Doodads (X3A – X3C).



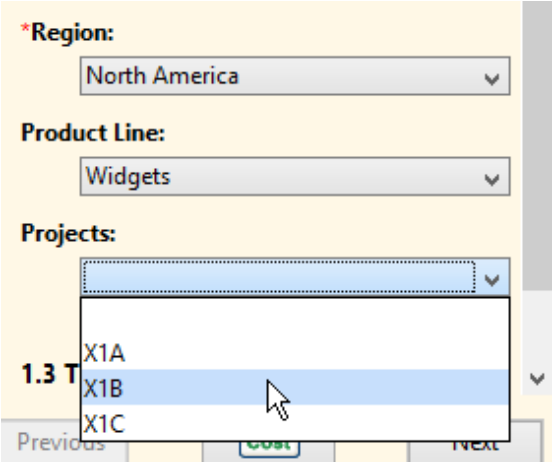
4 Make sure to make these UDAs visible with Dialog Views. (See *Managing dialog views* for more details.)

Publish your work, and then go to the Cost Guide and see the results:



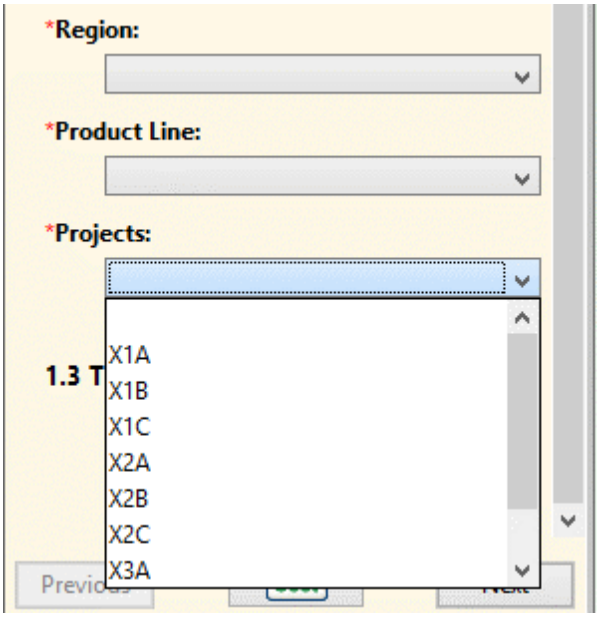
Assuming that you have not done any other customization, the new attributes will appear at the bottom of section **1.2 Company-Defined Attributes**.

Now when your users click through these attributes in order (top to bottom), each selection will determine what the next menu will offer. For example, selecting "**North America**" for the region ensures that only "**Widgets**" is available for **Product Line**, and only "**X1A, X1B, X1C**" are available for **Projects**:



Dependency Terminators

When we defined this simple example, we didn't take any precautions against having our users access these new UDAs out of order. Look what happens if they jump directly to **Projects** before selecting a **Region** and a **Product Line**:



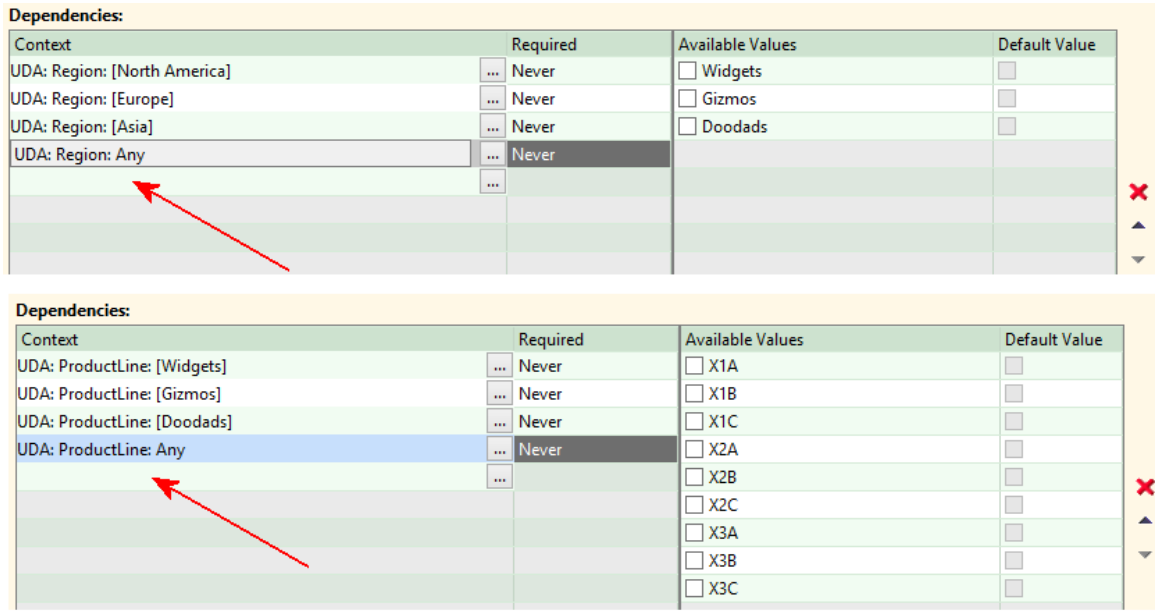
The **Projects** UDA drop-down menu will display a completely unfiltered menu (i.e., it displays all possible items). For this small example, this is not a huge problem: only nine items show up. In a real-life situation, the user might be overwhelmed with dozens or hundreds of options. And if they select one, and then select an item in the **Region** or **Product Line** that does not support that project, their original selection will be cleared and they will need to make another selection, this time from the valid options shown.

To avoid this kind of situation, we are going to add *dependency context terminators* to the **Product Line** and **Projects** definitions. This are special Dependency Context definitions that take effect if none of the expected conditions are met.

1 Go back to Steps 2 and 3 of the previous examples and add the following lines to the **Product Line** and **Projects** UDAs as shown in the screenshots below:

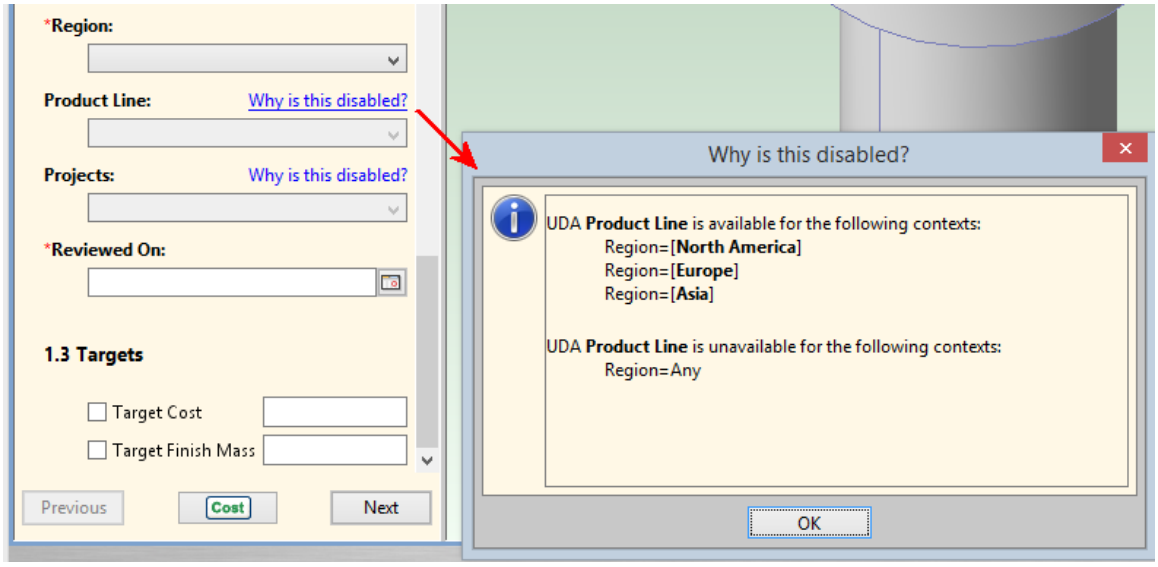
- UDA: Region: Any
- UDA: ProductLine: Any

(You specify “Any” by leaving the **Values** column blank when you define the Dependency Context.)



2 Re-publish, then bring up a fresh Cost Guide.

3 Now the **Product Line** field is disabled until you select a **Region**, and the **Projects** field is disabled until you select a **Product Line**. A “Why is this disabled” link explains to users how they need to proceed.



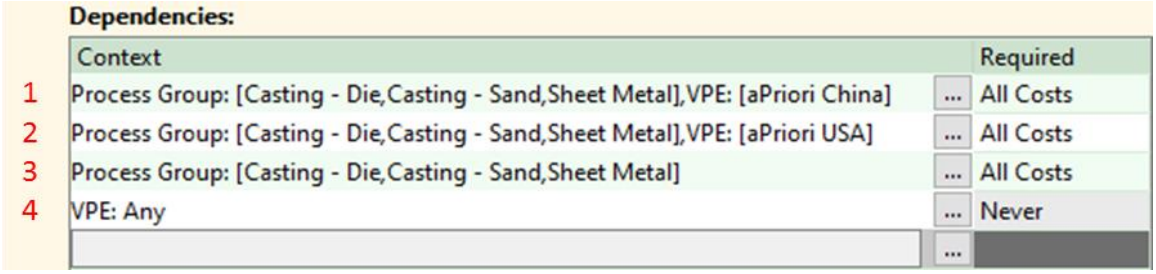
Dependency Context Details

The example in the previous section was very simple to get you started with Dependent UDAs. This section provides more details for more advanced use.

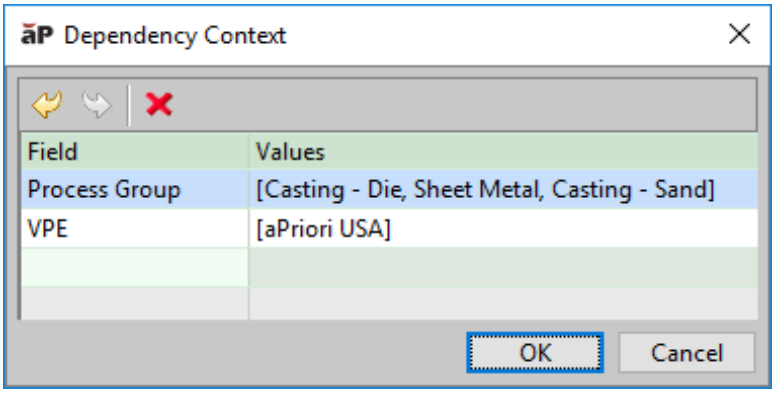
In addition to other UDAs, Dependency Contexts can be defined with Process Group, VPE, and Material selections.

There are a few important things to note about the logic within dependency definitions:

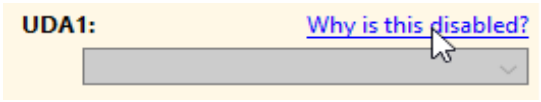
- The order of the Dependency Contexts is important, because the first context that is met will be used (1 through 4 in the screenshot below). It is recommended to have the most specific and highest priority Dependency Contexts at the top of the list.



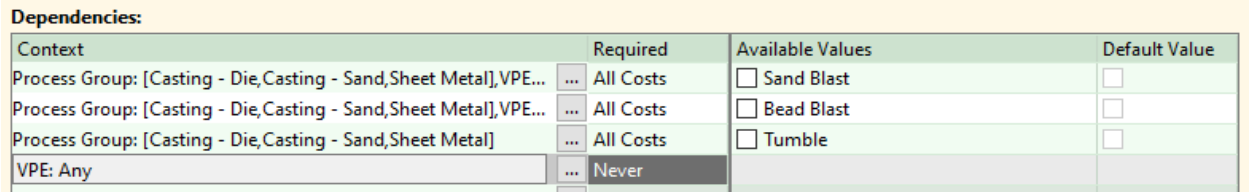
- Within a given Dependency Context, the logic is evaluated as "or" within a field and "and" from field to field. The context below would read as "if Process Group = [Casting – Die **or** Sheet Metal **or** Casting – Sand] **and** VPE = aPriori USA"



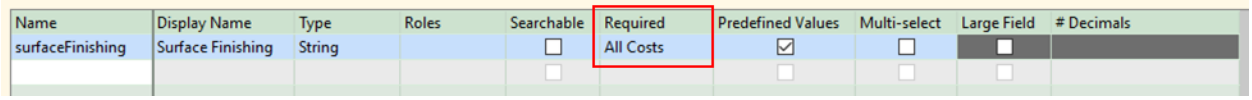
- If no values are made available for a given Dependency Context then the UDA will be rendered as read only in the user interface. Users will be able to access a dialog that explains why the UDA is disabled



- You should always define a "terminator context" – a general Dependency Context such as **VPE = Any**, which will evaluate to true whenever none of the Dependency Contexts above are met.



- If no Dependency Context is met, then all UDA level settings will be used – Required behavior, Default Value, and full Predefined List.



Name	Display Name	Type	Roles	Searchable	Required	Predefined Values	Multi-select	Large Field	# Decimals
surfaceFinishing	Surface Finishing	String		☐	All Costs	☑	☐	☐	

Predefined Values:

Value	Default Value
Sand Blast	☑
Bead Blast	☐
Tumble	☐

- You MUST ensure that UDAs are implemented in a logical order when configuring the **Cost Guide** view. Logical order is not enforced by aPriori.

Managing dialog views

Use the System Administrator **Dialog Views** window to specify:

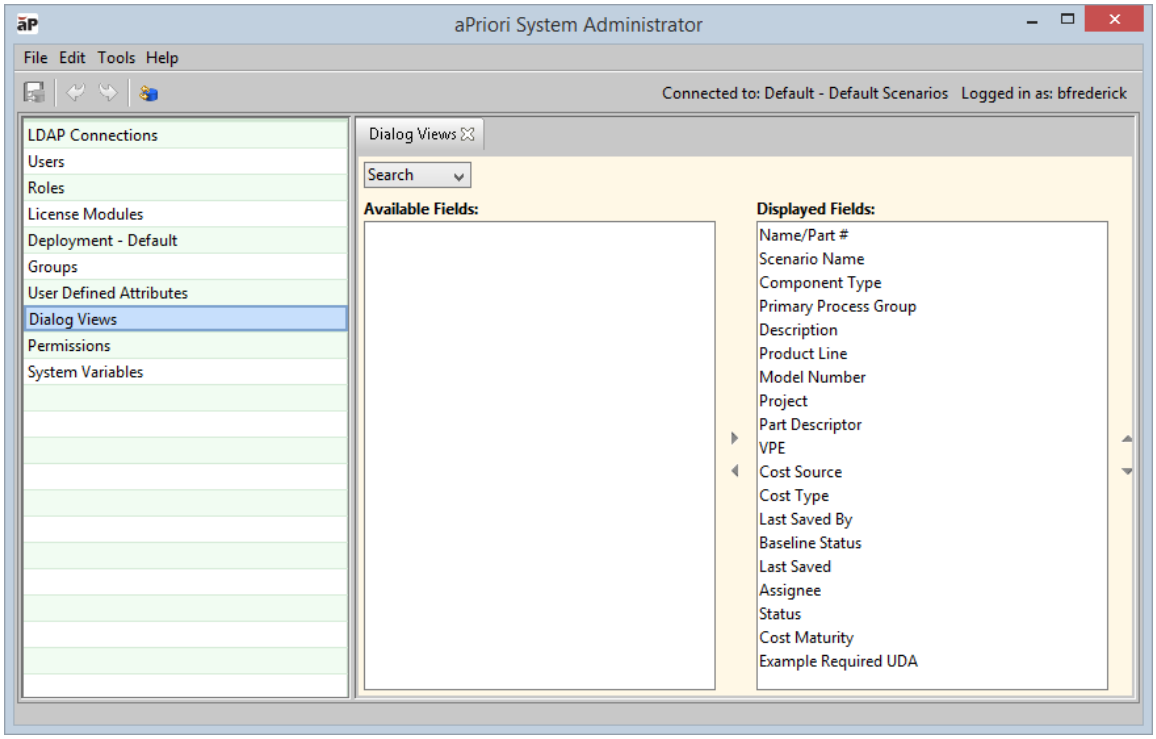
- Which searchable attributes appear on the **Search** window (also called the "**Search** tool"), and in what order.
- Organization of the first tab of the **Cost Guide**.

For more information about the **Search** tool, see "Using the Search tool" in the *aPriori Professional User Guide*.

For more information about the **Cost Guide**, see "Initial Costing: The Cost Guide" in the *aPriori Professional User Guide*.

To display the Dialog Views window

1. If the **System Administrator** window is not already open, click **Tools -> System Admin Toolset** from the main aPriori client, then click **System Administrator**.
2. Click **Dialog Views** in the Navigation pane to display the **Dialog Views** tab. (The **Available Fields** column will be blank if you have not customized **Displayed Fields** before.)



Controlling searchable fields in the Search tool

You can control which searchable attributes appear in the **Search Criteria** pane of the **Search** tool and where they are positioned. This allows you to show only the attributes that you want to make available and turning off the display of less useful attributes.

To display and position searchable fields in the Search window

1 If it is not already displayed, select **Search** from the drop-down list.

2 To hide a searchable field from the Search window, select it in the **Displayed Fields** column and click the left arrow (◄) to move it to the **Available Fields** column.

3 To display a hidden searchable field, select it in the **Available Fields** column and click the right arrow (►) to display it in the **Displayed Fields** list.

Note: You must display all required user defined attributes, or users will not be able to cost components.

4 Adjust the order of the displayed fields on the **Search** window by selecting a field in the **Displayed Fields** list and clicking ▲ or ▼ to move it within the list.

5 Select **File > Publish Changes** from the System Administrator menu bar or click  in the tool bar to save your changes.

6 Restart aPriori to display changes to the **Search** window.

Note: System Administrators at companies that have implemented aPriori Access Control can further affect the behavior of the **Search** tool by restricting the display of search results to only those components to which the user has access permission. See *Managing Systems* below for more information.

Customizing Fields on the Cost Guide

aPriori Professional Release 2018 R1 introduces several new options that allow System Administrators to customize the organization of the first tab (“Production Scenario”) of the **Cost Guide**. In previous releases, you could specify only the relative order of User-Defined Attribute (UDA) fields within the section “**1.2 Company-Defined Attributes**”.

This enables you to design a more efficient and user-friendly panel that meets company-specific practices and workflows. For example, if your company:

- has a User Defined Attribute that is directly related to the Process Group, then that UDA can be moved directly below the Process Group input.
- has many related inputs, then they can be put into their own group, with any name of your choosing.
- does not use target costs or target mass, then those inputs can be removed from the screen.
- always uses the material from the CAD model, then the material selection field could be removed.

The configuration capabilities include:

- Rename the Out-of-Box group headers (“1.1 Basic Options”, “1.2 Company-Defined Attributes”, and “1.3 Targets”).
- Remove group headers.
- Define new group headers.

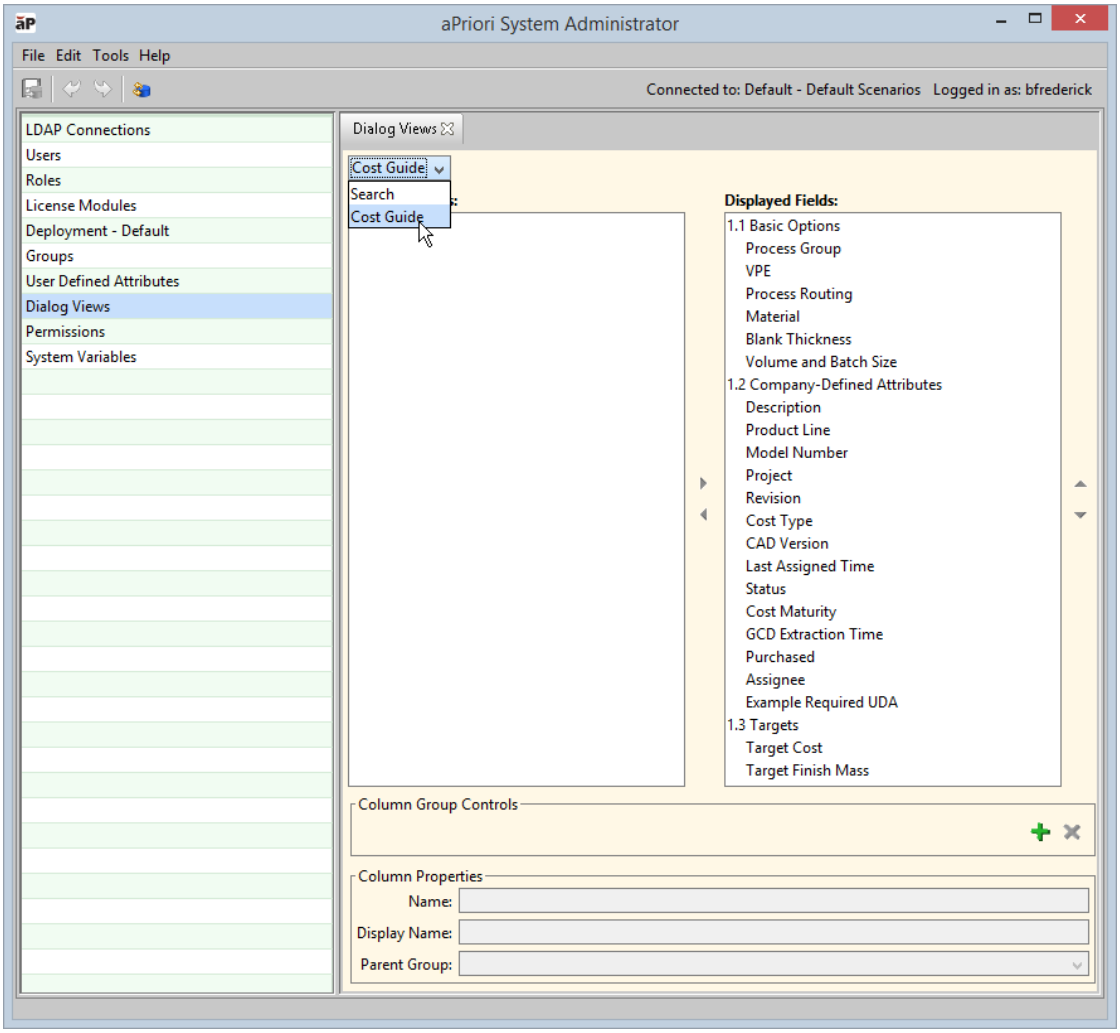
- Remove any fields (other than “Process Group” and “VPE”).
- Move fields to any location

Note: aPriori enforces that “Process Group” and “VPE” appear above their dependent inputs (“Material”, “Process Routing”, “Volume”, and “Batch Size”). The order of other dependent fields is not enforced. It is important to consider these dependencies when configuring the view to make the behavior as clear as possible to users.

Also note that changes made to the **Cost Guide** may cause changes to organization of the items on tabs in the **Cost Object Info** window. Fields will not change tabs in the **Cost Object Info** window: any attribute that exists by default in Cost Guide sections 1.1 (Basic Options) or 1.3 (Targets) out of the box will always appear on the Cost Object Info **Production Info** tab, and anything that appears by default in section Cost Guide 1.2 (Company-Defined Attributes) will continue to appear on the Cost Object Info **Administrative Info** tab. However, if you hide a field in the **Cost Guide**, it will also no longer display in the **Cost Object Info** window. And if you move a field up or down in the **Cost Guide**, its relative position to its neighbors will also change in the **Cost Object Info** window.

To customize fields on the first tab of the Cost Guide

1 On the **Dialog Views** window, select **Cost Guide** from the drop-down list. (Prior to Release 2018 R1, this option was labeled “Administrative Info” and provided limited functionality.)



2 **To rename a field or group header**, select it in the **Displayed Fields** column and edit the **Display Name** field in the **Column Properties** section at the bottom of the window.

Note: You should NOT change the display name of UDAs here, since the change will not be reflected elsewhere (such as in the BOM Loader, Bulk Costing, Part Details, etc.) You should only change UDA display names in the UDA definition. See *Managing User Defined Attributes (UDAs)* for details.

3 **To delete a group header**, select it in the **Displayed Fields** column and click the red "x" icon (✖) in the **Column Group Controls** section near the bottom of the window. Note that the deleted group's members still remain. You should manually move them to other groups.

4 **To hide fields or groups of fields,** select one or more fields and/or group headers in the **Displayed Fields** column (use the **Shift** and **Ctrl** keys to expand your selection). Note that you cannot select "Process Group: or "VPE". Click the left arrow (◂) in the center of the window to move the item(s) out of the **Displayed Fields** column. Note that if you hide a group header, all of its members also are hidden.

5 **To define a new group header,** click the green "+" icon ( ) in the **Column Group Controls** section near the bottom of the window. Enter a name in the resulting **Add Group** dialog box, then click **OK**.

6 **To move fields or groups** within the Cost Guide, select one or more fields and/or group headers in the **Displayed Fields** column (use the **Shift** and **Ctrl** keys to expand your selection). Click the up and down arrows (,) at the right of the column to move the item(s). Group headers (and their members) will move above or below the adjacent group. Individual fields will move up or down within their current group. To move a field to a different group, select the target group from the **Parent Group** drop-down menu at the bottom of the window.

7 When done, select **File > Publish Changes** from the System Administrator menu bar or click in the tool bar to save your changes. You do not need to restart aPriori to see your changes.

Managing permissions

Permissions are a key feature of aPriori Access Control. For information about this topic see *Permissions* in the Access Control chapter.

Managing System Variables – Search

The **System Variables** section contains two search-related variables:

- **enforcePermissionCheckForScenarioSearch** – aPriori 2018 R1 introduced the ability to restrict search results ONLY to those components to which the user has access. This only applies if your site has implemented Access Control. This setting is global.

  To ensure that the **Search** window (see "Using the Search tool" in Chapter 4 of the *aPriori Professional User Guide*) does not display results to which the user does not have access:

  - In the **System Administrator** window. click **System Variables.**
  - In the **enforcePermissionCheckForScenarioSearch** row, set **Value** to **true**. (The default is **false**.)

  Note: The default setting for this system variable is "false", which means that all search results are displayed–the same behavior that is found in all releases prior to 2018 R1. You must set this value to "true" as described above to turn on access-restricted search results.

  Administrators can further customize the behavior of the **Search** tool as described in *Managing dialog views* above.

- **includeRemovedUsersInSearchCriteria –** aPriori 2018 R3 SP1 introduced the ability to remove users without deleting their records from the database. To support this new functionality, **includeRemovedUsersInSearchCriteria** is provided to control whether or not removed users appear in search-related UI displays. The default value for this variable is **false**.

Managing Systems Variables – Other

The **System Variables** section contains three variables that are used for special preview releases of aPriori Cost Insight Design. In general, you should not modify the values of these settings except in specific circumstances related to CI Design installations. Contact aPriori Professional Services for any question about these settings.

3 Migration Import Tool

The Migration Import tool helps you migrate your data when you upgrade to a new version of the aPriori application. aPriori Support is available to help with this process and can provide a reference document for software upgrades and data migration. For more information, please contact aPriori Support.

4 Scenario Synchronization (deprecated version)

NOTE: *The command-line version of Scenario Synchronization is deprecated and is no longer supported. It has been replaced by the new Enterprise Platform web UI-based version. You cannot use both of these versions in the same environment, so if you are still using the command line version, please plan to move to the new version as soon as possible. Contact aPriori Support or Professional Services for more information.*

5 Access Control

aPriori's Access Control mechanism gives administrators the ability to configure role-based rules to define the access that members of groups have to components, assemblies, roll-ups, VPEs, and other resources.

This chapter includes the following topics:

Introduction to Access Control

aPriori provides an Access Control feature that allows administrators to define which users can perform which actions on what object types (resources). This means that you can define a security model that reflects your organization's needs, so that access is permitted or denied based upon corporate structures such as regions, divisions, projects, etc.

Note: aPriori strongly recommends that you contact aPriori Professional Services for assistance in creating and deploying an Access Control environment for your site. You can use the information in this chapter to learn Access Control basics in a test environment, but Professional Services will be able to help you develop and implement an Access Control mode customized for your production environment.

This chapter explains how to configure Access Control. There is also a short chapter in the aPriori User Guide that explains Access Control at a very high level for your end users who work in the Access Control environment that you configure. The User Guide chapter also contains information that administrators will find helpful regarding day-to-day use of Access Control, and how to interpret error messages and log files to troubleshoot Access Control issues.

As an aPriori administrator, you can:

- Provide granular access to different types of resources such as components, roll-ups, and VPEs.
- Control the actions that users can perform on these resources, such as create, read, update, delete, or cost.
- Organize users into groups and sub-groups and assign access control permissions to those groups.

In previous releases, aPriori has provided rudimentary access control of component and VPE data. This has been implemented through the mechanism of distinct component database schemas and distinct VPE schemas, which allow you to broadly segregate users and the components they cost, as well as the VPEs used to cost them. This is useful if you have multiple divisions which should not share information, or which are required to segregate products and components due to regulations. But as of aPriori Release 2015 R1, the Access Control feature provides much finer-grained control of groups of users to resources such as VPEs, parts, assemblies, and the different types of roll-ups.

Note: By default, Access Control for a new or upgraded installation of aPriori is configured to emulate the legacy approach of distinct component and VPE schemas. Sites that choose not to implement Access Control should see little or no difference in behavior after installing or upgrading.

Basic Concepts

In very simple terms, you implement aPriori Access Control by defining *user groups* that have *permissions* which consist of *rules* regarding the *actions* that can or cannot be performed on specific *resources* .

aPriori Access Control does not control access to fields or attributes, only to resources. However, rules can make use of custom attributes to fine-tune access to resources.

Note: For discussion purposes, Access Control definitions in this section are shown as typed statements. However, you typically will be defining Access Control using the aPriori GUI so do not be too concerned about the syntax shown. aPriori will construct the Access Control rules from your input. For more information, see *Using the Access Control UI* later in this chapter.

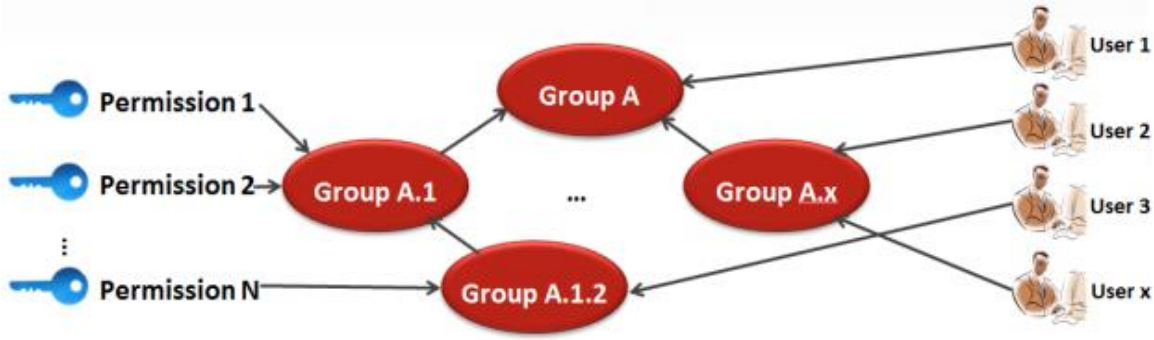
To expand on the terms introduced above:

Groups are collections of aPriori users. Groups can have subgroups. (Groups do not need to have users as immediate members; they can have only sub-group members, where users are members of the sub-groups.)

Permissions are associated with groups and consist of *rules* about the *actions* that can be granted or denied for a *resource*.

Resources are the entities to which you want to control access.

Rules consisting of *subjects* and *properties* can fine-tune conditions. *Rules* can be as simple or as complex as required, and you can use Booleans (AND OR) to chain rule segments together. (But when starting out, it is important to keep things simple.)



Resources

Access Control resources consist of the following:

- components (parts and assemblies)
- roll-ups (basic roll-ups, dynamic roll-ups, and cost comparisons)
- VPEs
- groups and permissions (these are used in advanced configurations and will be discussed later)

Groups

An aPriori group is a-named container of aPriori users (or sub-groups) who share the same access requirements.

Groups have permissions, which are discussed in the next section.

Groups can have sub-groups. You can re-use the same name for different groups; uniqueness is determined by the full path of each group. For example, "/USA/Devel" and "/EU/Devel" are both recognized as valid, unique groups. Every user (even administrators) belong to at least one group: the system-defined group "All Users". (Since slashes are used to specify group paths, you cannot create a group with a slash as parts of its name.) System Administrators can also assign users to additional groups as necessary to control the permissions they have. Note that groups are the primary mechanism for modeling your organization's access requirements.

There are four system-defined groups, including one sub-group. The names in parentheses show the internal aPriori name for the group, which you may encounter in different contexts:

- All Users (all_users)
- System Admins (administrators)
  - Super Users (super_user)
- VPE Admins (vpe_administrators)

These system-defined groups cannot be deleted, though the permissions associated with them can be edited. Note that all users (including members of the "System Admins" and "VPE Admins" groups) are members of the "All Users" group.

All other groups are defined by an administrator or other user who is granted group creation permission.

Groups have a default attribute ("name"), and a membership setting (None, Automated, or Manual), but can also have other custom attributes (not to be confused with User Defined Attributes).

If a user is a member of two groups, one of which allows access to a resource, and a second group that denies access to that same resource, that user will be denied access (although aPriori provides the ability to ignore or override "deny" permissions, which will be discussed later).

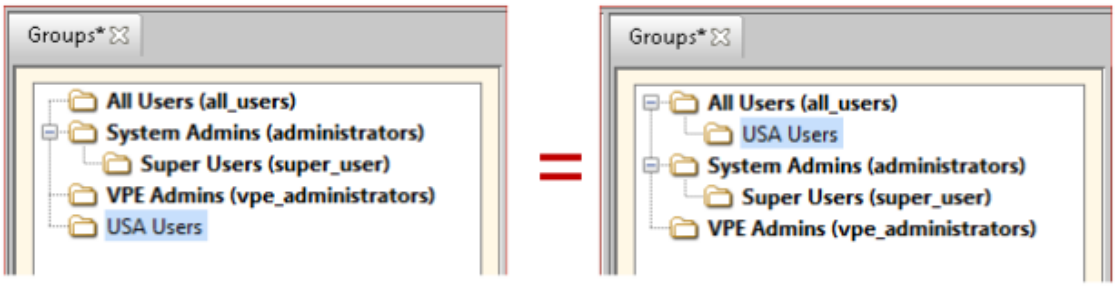
Note: If you make use of LDAP connections, see *Notes about LDAP Connections and Access Control Groups* in the System Administrator chapter.

Sub-groups

Groups can contain sub-groups, and sub-groups can also contain other sub-groups. Adding a user to a group automatically adds the user to any parent groups. Removing a user from a group automatically removes the user from all sub-groups. (Removed users are not removed from parent groups since they may belong to other sub-groups of a common parent.)

Sub-groups allow you to model access control situations where, for example, a given user is an administrative user for one project but just a "regular" user for a different project. In a case like this, the user could belong to the "Project A Admins" and "Project B Team Members" groups.

Note: There is not any real difference between creating a user-defined group at a peer level with the system-defined groups and creating it as a sub-group of All Users.



Super Users

The System Admins group has a sub-group called "Super Users", which automatically gets created (but not populated) at installation time.

Note: It is important that the administrator installing aPriori add himself or herself to the Super Users group before doing any other Access Control configuration. Once the Super Users group has at least one member, you can never accidentally delete the last member and potentially lock yourself out of the system.

As you might expect, members of the Access Control Super-Users group have powerful permissions that are not granted to other users. Specifically, they have "strongGrant" permissions for groups and permissions (described later in this chapter). In other words, they have Access Control powers over Access Control features.

Only the following Super-User characteristics can be changed, and they must be changed by a Super User:

- password (which you should change as soon as the installation is done)
- membership in the Super-Users sub-group (you can add and remove members)

Note that you cannot remove the last member of the Super Users group. If you need to remove the last user, you must first add another Super User.

Permissions

Permissions are the key to aPriori access control. A permission is associated with a group, and consists of:

- one or more Actions
- a Resource
- a Rule

In short, a permission specifies whether a member of the group to which it belongs is granted or denied the ability to perform an action on the specified resource, based on the specified rule.

A permission can also include Grant/Deny *behavior modifiers*:

- Normal Grant: Actions are allowed if the permission rule is True unless blocked by a Strong Deny permission

- Strong Grant: Actions are ALWAYS permitted if the permission rule is True
- Normal Deny: Actions are blocked if the permission rule is False unless allowed by a Normal Grant permission
- Strong Deny: Actions are blocked if the permission rule is False unless allowed by a Strong Grant permission

Actions

aPriori Access Control supports the following actions. Some are self-explanatory, but some require a little background.

- Create (available to all resource, but cannot be combined with other actions)
- Read (available for all resources except for groups and permissions; Read access is already granted by the system for every user to every group and permission to enable Access Control to work)
- Update (available for all resources)
- Delete (available for all resources)
- Cost Using (available for only VPE resources)
- Associate (available for only Permission resources)

Note that Create can be applied to all resources but cannot be combined with any other action.

The other actions may or may not be applicable to various resources but can be combined with other actions that have compatible requirements. For example, Update and Delete require that the Resource already exists, and can be combined in the same Permission. But Create requires that the target Resource does NOT already exist, and therefore cannot be combined with other actions which all require that the target Resource exists.

Similarly, you can create a Permission for Roll-ups that combines any combination of Read, Update, or Delete. But you cannot create a Roll-up permission with a Cost Using action, because Cost Using applies only to VPE resources.

Note: Read and Cost-Using are similar but have somewhat different behavior. In general. if you do not have Read permission for a VPE, you cannot see it. If you do have Read permission, you can see it and read the data in it, but you will not be able to execute a cost operation using it unless you have Cost-Using permission. There are some variations to this rule depending on context: For a user to see a VPE in a Cost Guide drop-down menu, that user must have both Read and Cost Using permission to that VPE. But in the VPE Manager, one needs only Read permission to see the VPE.

In the case of overlay VPEs, the top-level VPE is the only one that requires Cost-Using access. The underlying ("antecedent") VPEs do not require any access.

Resource

A Resource is the target on which the Action operates. As mentioned above, Resources include:

- Component (part and assemblies)

- Rollup (Roll-ups, Dynamic Roll-ups, Cost Comparisons)
- VPE
- Group
- Permission

Rules can use the attributes of a target Resource to fine-tune their behavior (see below).

Note: Group and Permission are provided for "access control of access control", which is an advanced topic. (See Access Control of Access Control)

Rule

A Rule is an expression that determines a group member's access to a target Resource. For example:

```
Group.attributesValue.Region == currentGroup.attributesValue.Region
```

If a rule evaluates to "true", then access is granted. If it evaluates to false, access is denied. However, there are grant and deny modifiers that determine how conflicting permissions are resolved.

Grant/Deny modifiers

The **Grant** and **Deny** permission modifiers give you the ability to fine-tune permissions by strengthening ("strong") access and denial results.

For example, permissions for Super Users who need exceptional access rights over access control feature will have the strong Grant modifier enabled. This means that Super Users are always granted access rights, even if another permission attempts to deny access.

Deny: Normal Deny is the default, Out-of-Box setting for "regular" user permissions and translates to "An action is blocked unless another permission explicitly allows it". In general, when you (an administrator) create a new permission, you should use Normal Deny unless a Strong Deny is required.

This is typically paired with a normal Grant setting (see below). A "normal Deny" is actually not a Deny at all–it is more of an "abstain" setting. This means that if a user is granted access by a permission, other permissions that return a normal Deny are ignored. Only a strong Deny permission will prohibit a user from an action form which they have been granted normal access. A normal Deny that abstains does not equate to granting access, it just means that the user will not be denied access if another permission grants it. This ensures that a user who belongs to multiple groups is not blocked from an action in one group because of a Deny in another group.

For example, assume that multiple groups are defined to correspond to projects. Each project has a Permission associated with it that has a component UPDATE action. The rule for each project might be something like:

```
component.projectName == currentGroup.attributeValues.Project
```

Now consider a user who is a member of several of these groups.

Without the normal Deny ("abstain") modifier, this user would not be allowed to update ("save") any scenario, because strong Deny permissions associated with the other projects would prevent it.

By setting normal Deny rather than Strong Deny on these permissions, failure of the rule is treated as if the permission did not exist in the first place. This is not a security hole, since one needs to have at least one Permission that explicitly returns True (i.e., grants access).

Grant – Strong: This modifier can force access to be granted even if there are other permissions that strongly deny access. This is intended for the Super Users group whose members may also be in other groups that have more restrictive permissions. This modifier should be used carefully and sparingly outside of the Super Users group.

The following table explains how combinations of these modifiers affect access and deny permissions:

Grant Level	Deny Level	Description
Normal	Normal	Access granted unless some other permission contains a Strong Deny. Access denied only if no other permission grants it. **NOTE:** This is the out-of-box default for "regular" users.
Strong	Normal	Access granted regardless of other permissions. Access denied only if no other permission grants it. **NOTE:** This is a superuser-level permission. Apply it with caution.
Normal	Strong	Access granted unless some other permission contains a Strong Deny. Access denied unless overridden by a Strong Grant
Strong	Strong	Access granted regardless of other permissions. Access denied unless overridden by a Strong Grant. **NOTE:** This is a superuser-level permission. Apply it with caution.

A Note about Terminology

Access Control terminology has evolved over time. It is possible that the terms and GUI labels that you see in the product today might be different than names that you have previously encountered.

Here is a mapping in case you encounter an Access Control term that you don't recognize:

Access Control Term:	May also be referred to as:
Resource	**target, targetString**
Action	**verb**
Rule	**predicate**
normalDeny	**abstain; softFail**

strongGrant	**override, nonOverridable**
Associate	**"Add to Group"**

Import and Export

Access Control provides the ability to import and export an Access Control model. You typically use this in one of two ways:

- to transfer a new or updated Access Control configuration from your development & test environment to your production environment.
- to transfer Access Control models between your environment and aPriori Customer Support or Professional Services.

The exported Access Control configuration consists of two parts which can be exported and imported through the UI.

- Access Control Objects – These include permission definitions, attribute definitions (name/type pair), groups and subgroups hierarchy, group attributes (name/value pair), and group permissions. This is saved as an XML file and *is not* user-editable.
- User-Group Associations – This is the list of users who belong to each group. This is saved as a spreadsheet .xls file and *is* user-editable. Groups are represented as a full path (for example, "administrators/managers". (This does not export users, nor does it define new users on import.)

Note: You must be a member of the Super Users group to perform Access Control import or export operations.

Importing Guidelines

When you import an Access Control model, the following behaviors are observed:

- Access Control Objects - All Groups and Permissions. Permissions imported into the target system completely replace (overwrite) the existing Permissions definitions in the target system. When importing Groups, aPriori attempts to maintain existing User-Group associations in the target system by updating existing Groups rather than by deleting and recreating them. Any groups in the target system that are not in the Import file are deleted.
- User-Group Associations - Complete overwrite. Importing a User-Group mapping completely replaces the existing User-Group associations in the target system. aPriori ensures that Users and Groups represented in the mapping exist in the target system. If any are found that do not, the import follows the 'log error and continue' model.

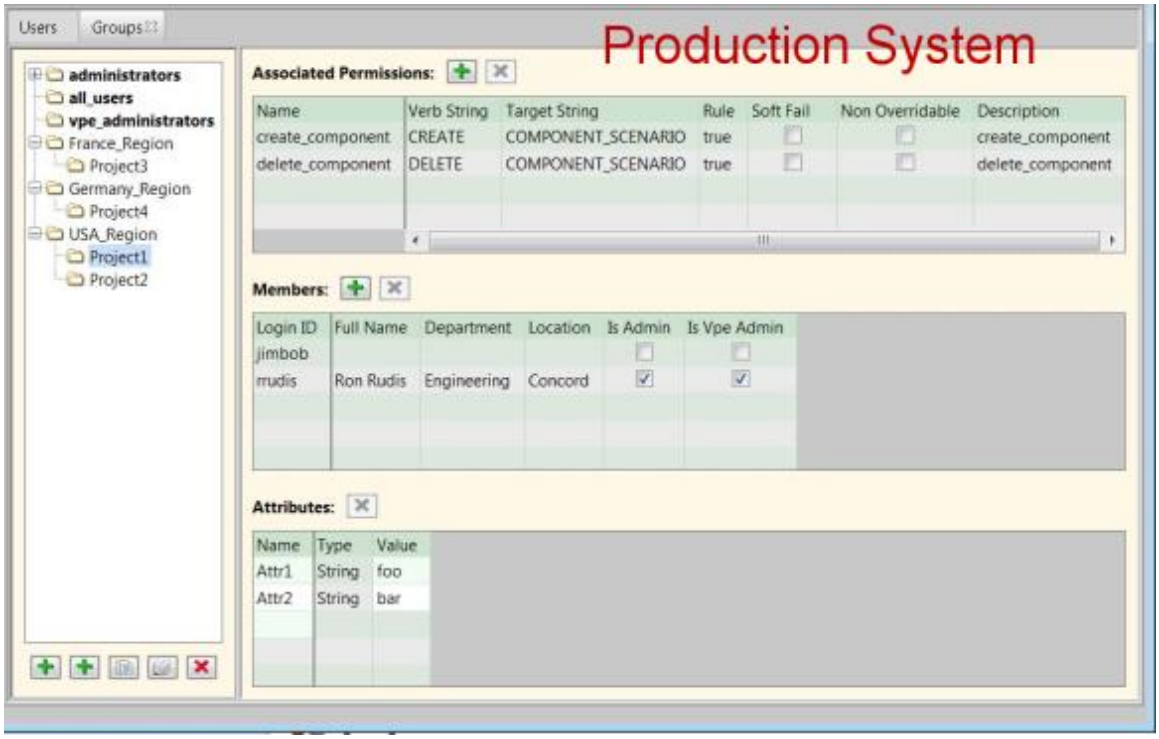
Note: A User-Group Association import replaces all existing associations on the target system, with the single exception that the last superuser will never be removed from the target system. Therefore, it is possible for an import to fail if it is started by a superuser on the target system who does not exist in the source system, and who is not the last superuser on the target system. (I.e.,

the superuser is removed by the import, and the import fails because that user no longer exists in the superuser group.) This also means that the source system can never be a "sub-set" of the target system, since importing it will cause all users not in the sub-set to be removed from their groups.

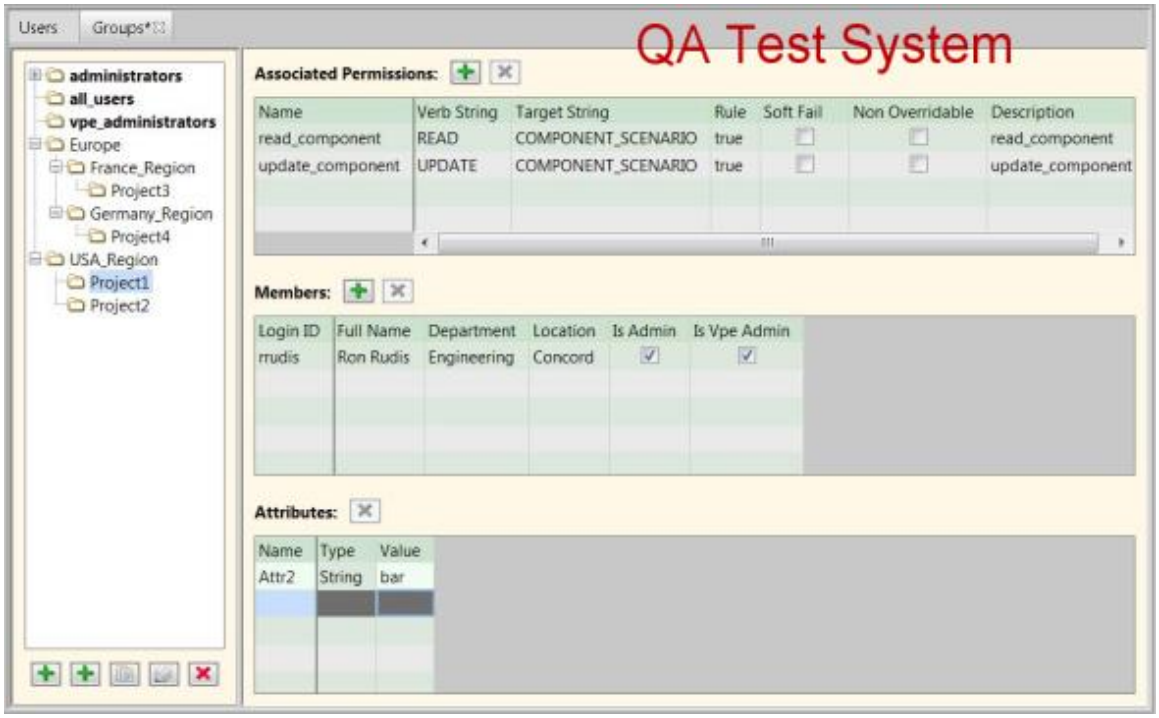
Importing Access Control Objects

When you import Access Control Objects, aPriori tries to maintain existing User-Group associations by overwriting existing Groups where the full path of the group is the same in the Import artifact and the target system.

For example, assume that a production system has the following configuration. Note that there are three top-level, user-defined groups: France_Region, Germany_Region, and USA_Region:



A new Access Control configuration with a revised group structure has been developed and tested on a QA machine. Now it needs to be migrated to the production system using Export and Import. Note that, among other changes, the two region groups are now sub-groups under a new user-defined "Europe" group.



After exporting the Access Control Object from the QA system and importing it into the production system, the states of the Group objects are:

- The **administrators** , **super_user** , **all_users** and **vpe_administrators** groups in the production system have the same Permissions and Attributes as they did in the QA system. The *members* for these groups are unchanged from what they were prior to the import.
- The **USA_Region**, **USA_Region\Project1**, and **USA_Region\Project2** Groups in the production system have the same Permissions and Attributes as they did in the QA system. The *members* for these groups are unchanged (they do not match the members from the QA system). In particular for the **USA_Region\Project1** group detailed in the screen shots above, this group in the production system after import will:
    - have the read_component and update_component Permissions.
    - not have the create_component and delete_component Permissions.
    - have the Attr2 attribute but not the Attr1 attribute.
    - have both jimbob and rrudis as Members.
  - The France_Region, France_Region\Project3, Germany_Region, and Germany_Region\Project4 Groups will be deleted from the production system.
  - The following new Groups will be created. They will have the Permissions and Attributes that they had in the QA system, and they will have no Members:
    - Europe
    - Europe\France_Region
    - Europe\ France_Region\Project3
    - Europe\ Germany_Region

- Europe\ Germany_Region\Project4

Best Practices

When using Export/Import to migrate Access Control models between environments, aPriori recommends observing the following guidelines.

- Whenever performing operations that will overwrite existing data, **make sure to back up your data first**.
- When making modifications and additions to your Access Control configuration, do so in a development & test environment, and deploy it to your production system only after extensive testing. Once the configuration is fully tested, you can migrate the Access Control configuration to the production system.
- When using a development & test system to modify your Access Control configuration, be sure to first export the Access Control configuration of your production environment and import it into your development &test environment. Using the same configuration data in all environments ensures that:
  - All environments will operate on the same baseline set of Access Control configurations as a starting point.
  - Any test configuration or access control items in a test or development environment will be removed, preventing those items from being migrated inadvertently to the production system.
- When importing an Access Control configuration to a production system, make sure to do it during planned downtime when end users are not accessing the production environment. Security issues could arise if an end user is using aPriori while an Access Control configuration is being imported.

The following example shows the migration of an updated Access Control configuration from a customer test (QA) environment to their production systems:

1. Ensure that the test environment includes all the users from the production environment. (You can do this from scratch, or by creating a spreadsheet and importing it, or by importing from LDAP if you have an LDAP environment. See the User and LDAP management sections of "Chapter 2 System Administrator for more information.)
2. Export the Access Control Objects from the production system.
3. Export the User-Group Associations from the production system.
4. If your QA system contains data that you wish to retain, back it up.
5. Import the Access Control Objects into the QA system.
6. Implement and test changes on the QA system.
7. *[Optional]* Modify the User-Group Association export file from Step 2 to account for any Group hierarchy changes. (You can edit this file directly.)
8. Export Access Control Objects from the QA system.

9 Back up the production system.

10 Import the Access Control Objects artifact from the QA system into the production system.

11 *[Optional]* If you modified the User-Group Association artifact in Step 6, also import it into the production system.

Using Import/Export for Access Control

Exporting Access Control Objects

Access control related objects are exported in XML format and should not be edited manually.

To export access control related objects:

1 From the System Administrator window, click **File > Export > Access Control Objects**

2 In the resulting window specify a path and file name, then click **OK**.

Exporting User-Group Associations

User-Group Associations are exported in human-readable format and (unlike exported Access Control Objects) can be edited manually.

To export User-Group Associations:

1 From the System Administrator window, click **File > Export > User Group Associations**

2 In the resulting window specify a path and file name, then click **OK**.

Editing User-Group Associations

User-Group Associations are exported to a user-editable .xls spreadsheet file. The structure is similar to exported VPE data.

There is a sheetIndex sheet which has logical and Excel names columns. The logical name is a path to the group and the Excel name is an internal name of an Excel sheet. Due to length and character set limitations, the actual group path is not used directly as a sheet name.

Each group is represented by a sheet with the login IDs of currently-associated members.

To find a group, switch to sheetIndex, click on a target group path (paths in the logical name column are hyperlinks) and the sheet with its members is selected automatically.

To add a member to a group, type the login name.

To delete a member from a group, select the whole row and click Delete from the menu.

Note: When editing associations by adding or removing login IDs, take into account that:

- A user added into a subgroup is automatically added to all parent groups in the group's path.
- A user deleted from a group must also be deleted from any sub-groups. For example, if “user1” is a member of a parent group and a child group and you delete it from the parent, you must also delete it from the child group. Failing to delete the user from both groups will cause an error upon import.

An entire sheet can also be deleted in a workbook so that all existing users are preserved upon import (i.e., a group will be skipped).

Importing Access Control Objects

To import access-control related objects:

1. From the System Administrator window, click **File > Import > Access Control Objects**
2. In the resulting window, navigate to the desired Access Control Object export file (these always have an "aco.xml" extension), then click **OK**.

   The permissions and attribute definitions in the current system are replaced with those from the .xml file.

Groups and hierarchy, group permissions, and group attributes are synced by the group's path and reflect the exported data.

The four system-defined groups will not be deleted even if their definitions are removed from the .xml file:

- All Users (all_users)
- System Admins (administrators)
- VPE Admins (vpe_administrators)
- Super Users (super_user)

Only the associated permissions and group attributes for these groups will be updated.

You do not need to "**Publish Changes**" after importing.

If any Access Control-related windows are open in the System Administrator Toolset, they are updated to reflect new data.

Importing User-Group Associations

To import User-Group Associations:

1. From the System Administrator window, click **File > Import > User Group Associations**
2. In the resulting window, navigate to the desired User-Group Association export file (this will have a ".xls" extension), then click **OK**.

Upon import the all_users group is skipped so that no users can be added to or deleted from it.

You do not need to "**Publish Changes**" after importing.

If any Access Control-related windows are open in the System Administrator Toolset, they are updated to reflect new data.

Both imports automatically save imported data, so there is no need to publish changes in the System Administrator Tool UI. If an Access Control-related window is opened in the System Administrator Tool it is updated to show new data.

Guidelines and Principles for Access Control

Before diving into Access Control and trying to implement a complete model that addresses all the access requirements for your entire organization, we highly recommend that you:

- study the content of this section and get a good understanding of it before proceeding
- involve aPriori Professional Services in planning and implementing any Access Control model for your site.

When learning aPriori Access Control, perhaps the most important guideline to follow is to Keep It Simple. In a test environment, try implementing a single permission for a small group, and once it works the way you want, expand.

You should involve aPriori Professional Services early in the process of starting to develop an actual Access Control model that will be deployed at your site.

As with every new feature, you should master and validate this functionality in a test environment first and roll it out to your production environment only when you are sure that it works. Try rolling it out to a small pilot group of advanced users initially. Like most access control systems or powerful programming languages, aPriori Access Control does not prevent you from defining a rule that is not logical. Build up slowly so that you can debug your new Access Control rules more easily.

Access Control Principles

aPriori Access Control incorporates the following principles to protect your data and to provide security without being a burden to your users.

1. Users cannot grant themselves new permissions as a result of updating a resource. For example, users cannot change the value of an attribute to give themselves Update rights if they did not have those rights already.

2. Users cannot give away permissions that they had as a result of updating a resource. For example, if users have Read + Update rights, they cannot change the value of an attribute so that they can no longer Read or Update it.

3. Fail fast when possible: For example, in order to BOM load, users need to be able either to Create and Read and/or Update the parent roll-up and potentially any functional group roll-ups. aPriori checks these conditions up front and fails immediately if it finds a problem, before starting the BOM load operation.

Note: When a BOM load creates a new scenario for an existing part, it does so by copying from the existing part. If your site has implemented Access Control, in some situations this can cause the BOM load to fail due to an Access Control error – READ privileges are checked against the existing component, to which you may not have access.

4 Keep going and notify: Once past the initial "fail fast" checks, for operations that involve manipulation of multiple scenarios, aPriori skips over scenarios that fail due to Access Control but logs the issue and notifies the user. For example, after the initial check for roll-ups, the BOM loader loads everything it can and notes any access control failures.

5 Abort the entire operation: If any part of an Update operation fails due to Access Control, the entire operation is aborted and aPriori attempts to clean up (delete) any scenarios created up to that point. For example, Scenario > Save As > Scenario & Children could fail an Access Control check after already processing some number of scenarios. In this case, aPriori stops the save and cleans up any scenarios that were created. (Spreadsheet reports work in the same manner, except that there is no need to clean up.)

Note: "Fail fast when possible" does not apply in this situation for performance reasons. It would not be efficient to visit each scenario to check if all permissions are okay before performing the Save operations.

6 Check before destroy: aPriori checks Access Control permissions before deleting data. For example, using File > Import to import scenarios from an ".ap" file will delete any existing scenarios that conflict with the ones being imported. Before performing the delete, aPriori verifies that the importing user has the necessary permissions for the entire import to succeed.

Guidelines

In addition to the principles listed in the previous section, there are some basic Access Control rules to remember:

1 If you change the Access Control permissions for users who are logged into the system, the changes will not affect them until they log out and log back in again.

2 Users can belong to multiple groups. Every aPriori user belongs to at least one group: All Users. But most users also get assigned to one or more other groups that reflect something like the organization’s regions, or projects, or job function.

3 A user who belongs to a sub-group automatically belongs to all of that sub-group's parents, grandparents, etc.

4 To have access permission to a resource, a user must:

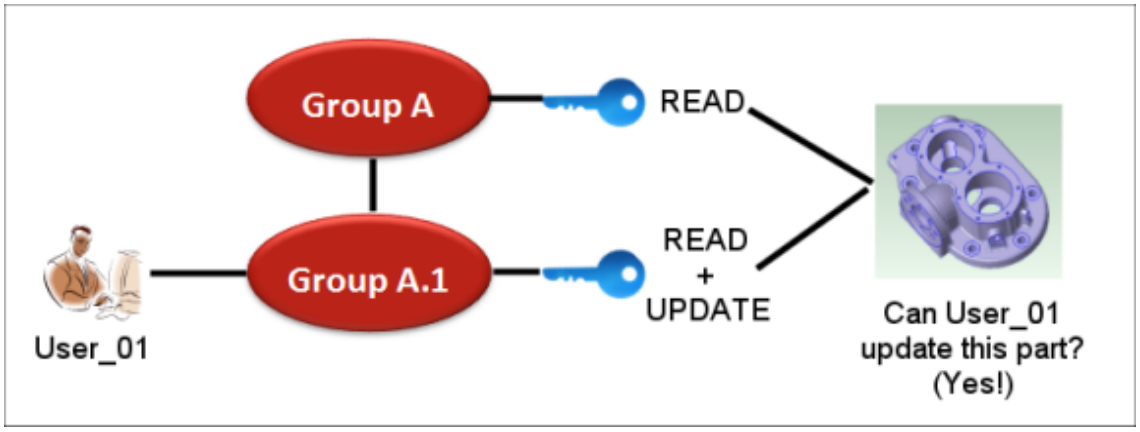
- belong to at least one group that grants that permission,
- NOT belong to any group that denies that permission with Strong Deny

5 The most restrictive permission is the one that counts. For example, a user might belong to one group that allows him or her to update a component, and to another

group that explicitly prohibits (via Strong Deny) updating that component. In this case, the "deny" permission takes precedence and the user cannot update the component.

There are two modifiers called "Grant" and "Deny" that allow you to fine-tune this behavior. "Grant/Deny modifiers" on page 72 for more information. Also note that a given group can have multiple permission definitions consisting of the same action and the same resource, but with different rules.

Note: *Group Inheritance vs. "Most Restrictive Permission Wins"* – As mentioned above, a user who is a member of a sub-group is also automatically a member of all of that sub-group's parents. So what happens if a user belongs to a sub-group that grants Read + Update permission to a component, but the parent group grants ONLY Read access? Does the more restrictive permission of the parent prevent the user from updating the component?



By default, the user is not *prevented* from updating the component. Why? Because the Read permission of the parent group does not explicitly deny Update access to the user, it simply does not grant update access to its members. So somebody who is a member of ONLY the parent group will not be able to update the component, but somebody who is a member of the sub-group can. The parent permission does not grant it, but it does not deny it either. Further note that there is nothing special about parent-child group relationships when it comes to permissions. The permissions for each group are evaluated independently for each group, regardless of whether they are peers, or parent-child, or completely unrelated.

Example Use Case

aPriori Access Control does not assume any particular use case by default. It is designed to be flexible so that you can configure it for your own needs.

The out-of-box default settings are meant to make Access Control invisible to upgrading aPriori users. That is, they are designed to emulate the way aPriori has traditionally worked prior to Release 2015 R1 and the introduction of Access Control.

If you decide to implement your own Access Control model, you will want to modify or remove the out-of-box settings and replace them with new ones that reflect your model. You should contact aPriori Professional Services for assistance developing a custom Access Control model for your site.

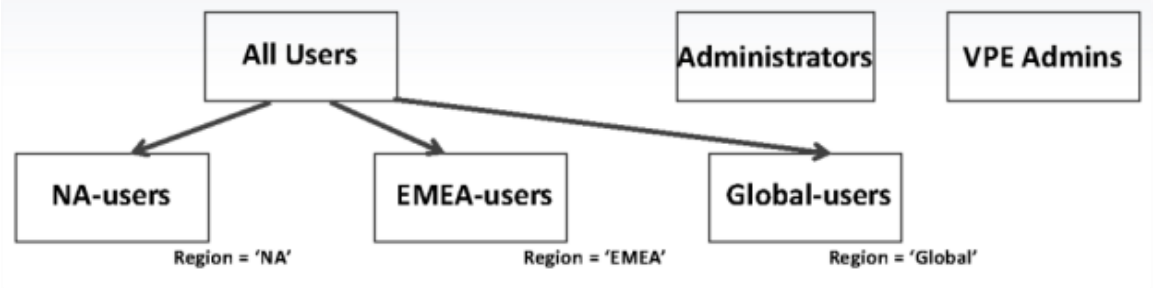
The following example environment is intended to give you an idea of how such a custom Access Control model could be designed and implemented.

Region-Based

Assume that you want to control access to resources based on a geographic breakdown of your organization. For example, none of the components or VPEs created in the North American (NA) region should be able to be read, updated, or deleted by anyone outside of the NA region.

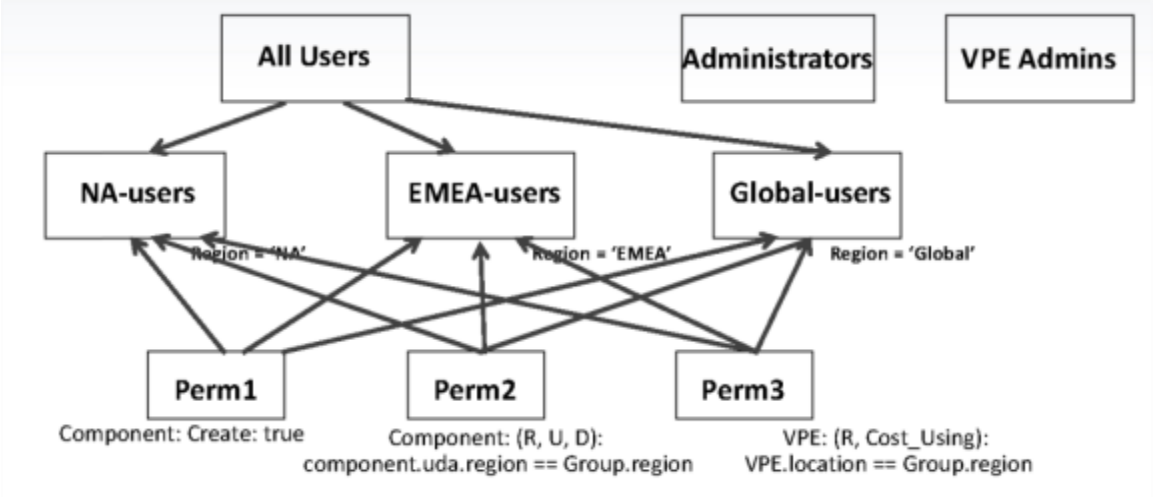
In this example, we assume that users can only belong to one region. However, if this is were not true, we could use modifiers to tweak the model.

Here are some suggested steps that you could follow to implement this access control model:



- Create groups that reflect your geographic divisions. For example, "NA-users", "EMEA-users", and "Global-users".
- Create a group attribute called “Region” and set it appropriately for each group (for example, "NA" for the "NA-users" group, "EMEA" for the "EMEA-users" group, etc.)
- Also create a similar "Region" UDA for your component scenarios, and plan to coordinate these with a VPE property such as Location or Description.
- Add your users to the appropriate groups.
- Set the "Region" UDA and the VPE region property to the appropriate values to coordinate with the "Region" group attribute. (For more information about UDAs and using them with components and assemblies, see Managing user defined attributes). You will be comparing these values within permission rules, so you must ensure that the value strings are identical.

Next create permissions and associate them with the groups:



- Perm1: Component: Create: true
- Perm2: Component: (R, U, D):
  component.customAttributes.region == currentGroup.attributeValues.region
- Perm3: VPE: (R, Cost_Using):
  VPE.location == currentGroup.attributeValues.region

A note about roll-ups

If you have roll-ups, you should implement a consistent roll-up naming strategy that incorporates the region into the name. (Roll-ups currently cannot make use of UDAs.) For example:

"NA_Rollup_XYZ", "EMEA_Rollup_XYZ", etc.

Your permission rules would then need to reflect this naming convention. For example:

Perm4: Rollup: (U, D, R):
index(upCase(rollup.name),upCase(currentGroup.attributeValues.region)==1

A Note about VPEs

In the simplest case, your regional groups might be the same as your VPE locations. For example, all North American users could have the same access to the aPrioriUSA VPE. But what if your North American region makes use of separate VPEs for USA and Mexico?

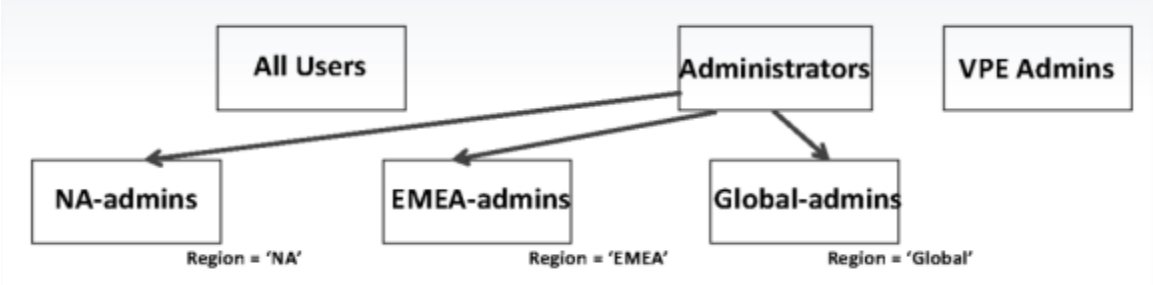
If VPE.location values are not the same as the Region values, you might consider using a different VPE property such as "Description", or even editing the default value of the "Location" property. "Location" is not used internally by aPriori, so if your site does not already depend on this property, you should be able to modify it as necessary.

You could then develop a rule such as:

VPE.location == currentGroup.attributeValues.Region

Configuring Administrators

Setting up access control for administrators in multiple regions can be handled in a manner similar to the users shown in the previous sections. (This configuration is not necessary if all of your administrators have access to all regions.)



- Create a group for administrators in each region, such as "NA-admins", "EMEA-admins", and "Global-admins".
- Add administrators to the appropriate groups.
- Assign the "Region" group attribute to each group and set it to reflect the region.
- Assign a VPE property to the same Region values, as described in the previous section.
- Create different rules and associate them with the groups. For example, you might give more access rights to VPEs for administrators.

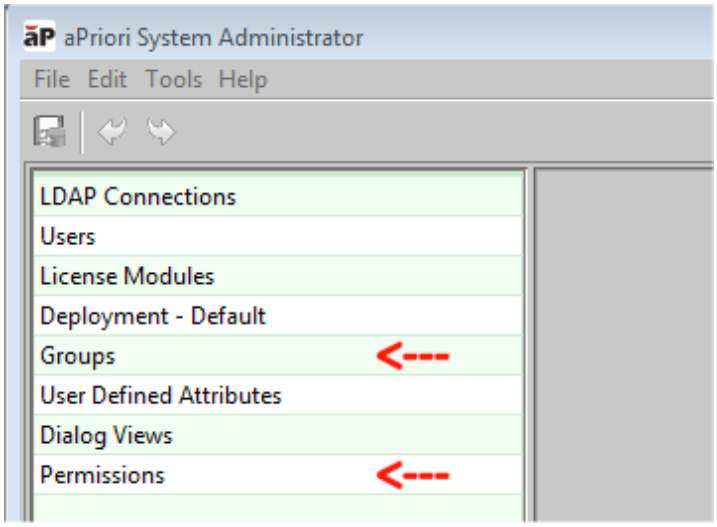
Extending the Example

You could extend the access control requirements of the previous example by adding project and role-based rules to the regions, and also adding life-cycle status. For assistance developing these kinds of Access Control models, please contact aPriori Professional Services.

Using the Access Control UI

The information presented to this point has been mostly conceptual and theoretical. So how do you actually implement these capabilities through the aPriori desktop?

You define Access Control from the System Administrator window, using the **Groups** and **Permissions** tabs.



You may also access the Users tab for some purposes (for example, you can assign users to groups from the **Users** tab as well as the **Groups** tab) but most Access Control configuration is done through **Groups** and **Permissions**.

The following sections are provided to get you oriented so you can see where the elements mentioned in the conceptual part of this chapter are made available through the UI.

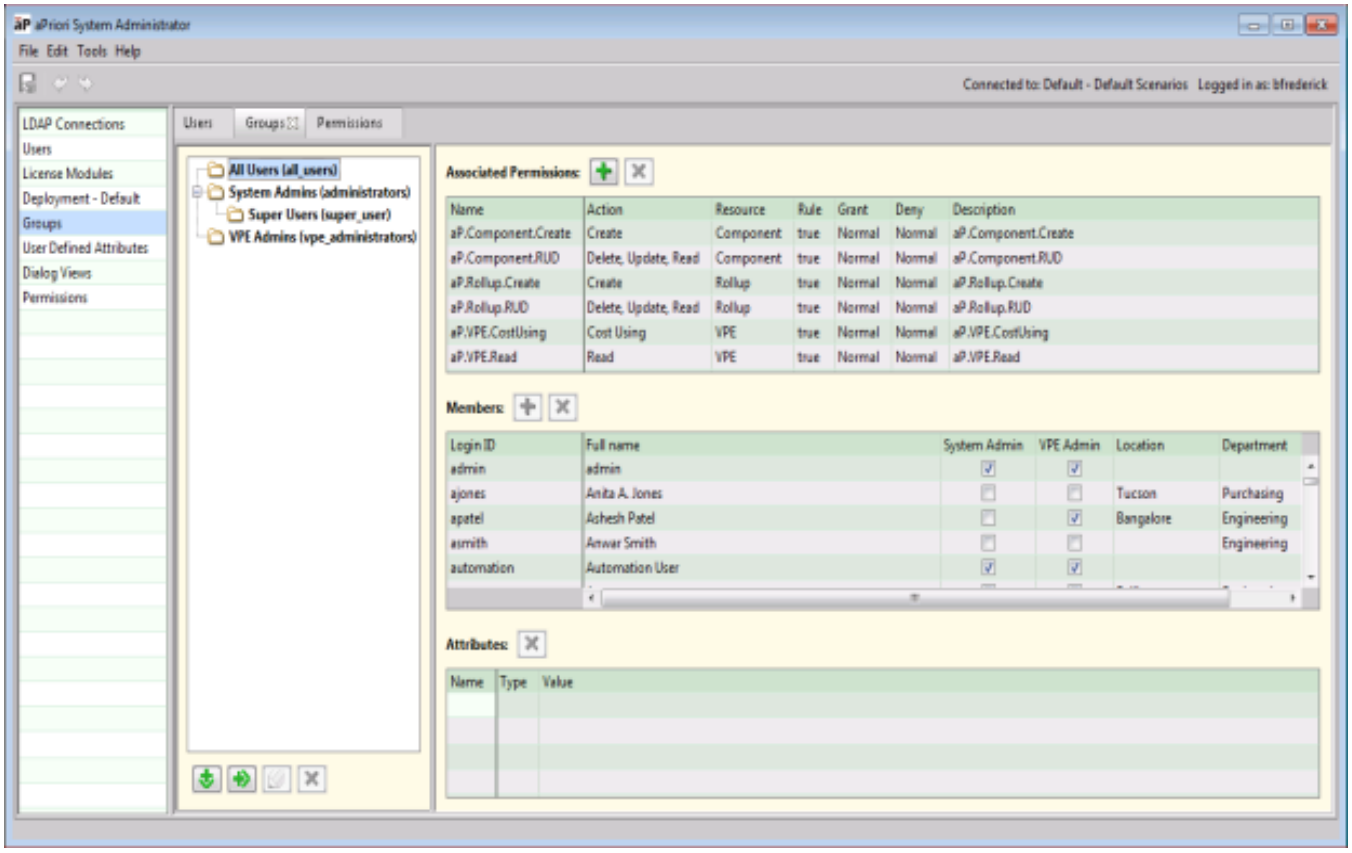
Groups Tab

Use the **Groups** tab to create or delete groups and subgroups, assign users to them, associate them with specific Access Control permissions, define group attributes, and specify group membership behavior. As mentioned in the conceptual discussion earlier in this chapter, Groups are key to modeling your organization and granting or denying access to various resources to several users.

Note: As of Release 2017 R1, there are two changes to group management:

- Group membership is now handled separately from any LDAP Maps, EXCEPT for the system defined special groups All Users, System Admins, and VPE Admins. All other user-defined groups must be provisioned manually.
- Group membership changes are handled by the Group Membership Process that automatically runs when you use the Publish command to commit your changes, or as the final process of an automated LDAP Synchronization job. The process runs only if a change has occurred on the Users, Groups, or Permissions tabs. It cannot be invoked or cancelled manually.

Here is how the Groups dialog might look at your site before any major Access Control configuration.



Out of the box, you will see four system-defined groups:

- All Users
- System Admins
- Super Users (a sub-group of System Admins)
- VPE Admins

The lower-case names in parenthesis are the system equivalent names that you would specify if importing/exporting groups via spreadsheet. ("Import and Export" on page 74).

These system-defined groups have the following default settings:

- All Users: no type (nothing is shown in the UI; it is handled automatically by aPriori; the UI does not allow users to be added or removed)
- System Admins, VPE Admins: Manual
- Super Users: Manual (and this cannot be changed).

When an admin edits the group membership type for an existing group and enters a different type, the following rules should apply.

- **Old Group type:** None, **New Group type:** Manual or Automated

  Action: do not change the current group membership (note that the next Group Membership Process invocation will fix up the membership if it needs to change)

- **Old Group type:** Manual, **New Group type:** None

  Action: Remove current members (who are not populated by a child group)

- **Old Group type:** Manual, **New Group type:** Automated

  Action: Do not change the current group membership (the next Group Membership Process invocation will fix up the membership).

- **Old Group type**: Automated; **New Group type:** None

Action: Remove current members (unless they are populated by a child group)

- **Old Group type**: Automated; **New Group type:** Manual

Action: Remove current members (unless they are populated by a child group)

To Add, Edit, or Remove groups or sub-groups

To add a new group, click the **New Group** button: 

To add a new sub-group, select an existing group and click the **New Sub-Group** button: 

To edit a group or sub-group, select it and click the **Edit Group** button: 

Note that you can only edit the name or the Group Membership setting with this button. To edit the Permissions, Members, or Attributes of a group or sub-group, select it and use the panes on the right side of the window. Further note that you cannot edit the All Users or Super Users groups at all, and only the Group Membership setting for System Admins and VPE Admins.

To remove a group or sub-group, select it and click the **Remove Group** button: 

Note that you cannot delete any of the four system-defined groups (All Users, System Admins, Super Users, or VPE Admins).

Group Membership settings

This setting determines how user are associated with groups:

- **None –** The group contains only users from sub-groups. You cannot add or remove users from directly through the UI.
- **Manual –** The group contains users who are assigned through the UI, or by importing user/group associations (see *Import and Export*).
- **Automated –** The group is populated by the Group Membership Process. These groups have a new memberOf permission associated with them.

Note that for LDAP connections to have the **System Admin** and **VPE Admin** options available for the **Sync Admin Group Membership** settings (see *To add a new LDAP connection*), these groups must be set to **Manual** here.

Associated Permissions, Members, and Attributes panes

The Associated Permissions pane displays the permissions that currently apply to the selected group. In the screen shot above, all permissions shown are the system-defined permissions available Out-of-Box , as identified by the "ap." naming prefix.

The Members pane displays the users who belong to the selected group.

The Attributes pane allows you to define, associate, and remove attribute/value pairs for the selected group. To create a new attribute, double-click the Name field for the first unpopulated row and enter a name for the attribute. Double-click the Type field to display a pull-down menu from which you can select **string** for text, **double** for numeric values, **Boolean** for true/false values, or **list** for semi-colon-separated strings. Double-click the Value field to enter the value of the attribute.

An example of using the list attribute would be to create a list of VPEs to which the group should have access, such as setting attribute "AC_VPEs" to:

aPriori China; aPriori USA; aPriori Mexico

This could then be used by an Access Control administrator to write a rule similar to:

```
vpe.name in currentGroup.attributeValues.AC_VPEs
```

Note: When entering or editing attribute values, you should always press the **Enter** key when done, and before clicking elsewhere in the UI. Otherwise, the value may not be saved.

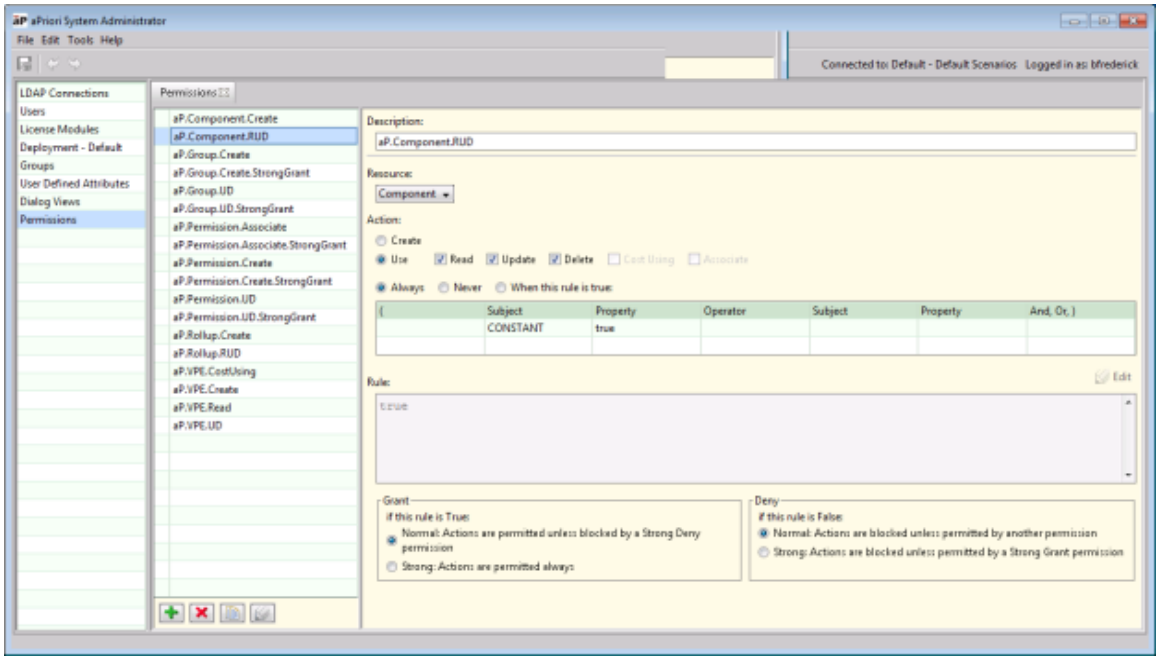
When done, click the **Publish** icon to save the new attribute and make it available to other groups.

UDAs and Group Attributes

Note that group attributes are separate from User Defined Attributes (UDAs) which apply only to component scenarios. However, you can manually coordinate group attributes with UDAs. For example, if you implement a region-based Access Control model, you might create both a group attribute and a UDA named "Region" and set them to values such as "NA" for "North America" or "EMEA" for "Europe, Middle East, and Africa". Once defined, you can then create permission rules that ensure that members of a group with the "Region" group attribute set to "NA" can only access component scenarios. You must ensure that you implement the names and values EXACTLY so that rule comparisons return valid results.

Permissions Tab

Use the Permissions tab to create, view, delete, or modify access rules. These permissions can then be associated with groups via the Groups tab (see previous section). The screen shot below shows how system-defined permissions appear Out-of-Box , before any customization work has been done.

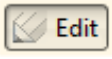


All aPriori-defined permissions have descriptive names starting with "aP." and indicating their purpose. For example, "aP.component.RUD" is a system-defined permission that gives the user Read, Update, and Delete rights to components. aPriori recommends that you define and implement a strict naming convention for any permissions that you develop for your site.

Note that you do not necessarily need to use a single prefix such as "user." for all of for your own rules: Just the fact that they do not begin with "ap." flags them as user-defined, so you can simply begin your names with categories or some other organizational prefix.

The controls at the bottom of the permissions list allow you to add, delete, copy, or edit permissions:

Control	Purpose
	Add – Create a new permission from scratch.
	Remove – Delete the selected permission.
	Copy – Make a copy of the selected permission, typically to use as a starting point for a new permission, or to make a back-up of a permission prior to editing. By default, the new permission is created with the same name as the source permission, but with "Copy" appended. aPriori displays a dialog so that you can customize the name of the copied permission.
	Edit – Make changes to an existing permission.

Note: When you finish entering or editing a rule, you should make sure to click the pencil **Edit** icon (

). If you try to go directly to the Publish icon, for example, you may get a message that no rule is defined, or your edits may not get saved.

To create a new permission

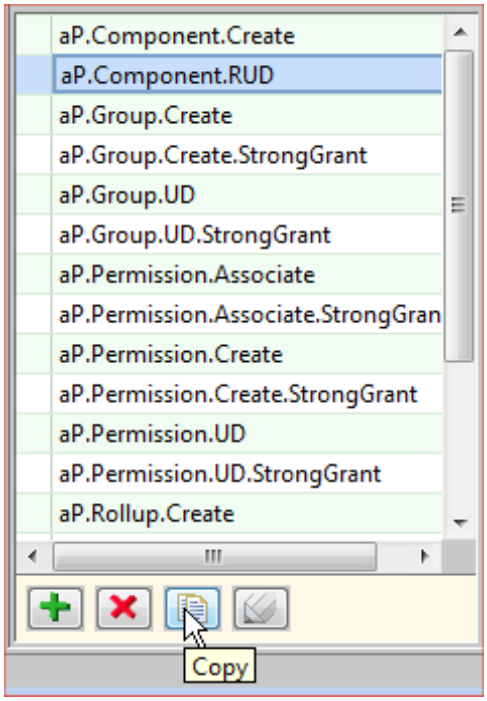
The general flow for creating a new permission follows the steps show below. Before creating new permissions, you should remember the following important guidelines:

- Develop, modify, and test permissions ONLY in a test environment. Do not deploy a new or modified permission in your production environment until you are 100% confident that it works the way you think it works.
- Keep it simple. Build up permissions incrementally and, again, test them thoroughly before deploying them to your users.
- The UI helps you build the framework of permissions with a "Rule Builder", where the elements of your rule are displayed in the Rule field. You can define simple rules without ever editing the Rule field. However, you can also build quite complex rules using the aPriori CSL (Costing Syntax Language) statements. For advanced implementations such as this, contact aPriori Professional Services, and watch for examples that aPriori will be developing over time.

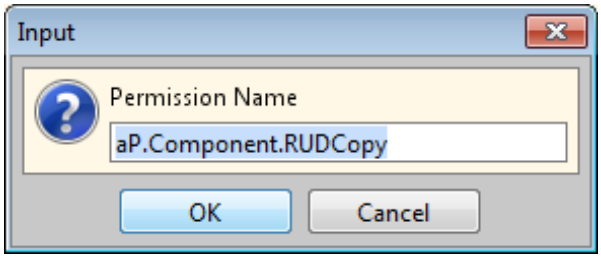
To create a new permission:

1 You can either start from scratch by clicking the plus-sign **Add** icon or clone an existing permission to use as a starting point. It is often more efficient to find a rule that is similar to what you want, copy it, and modify the copy.

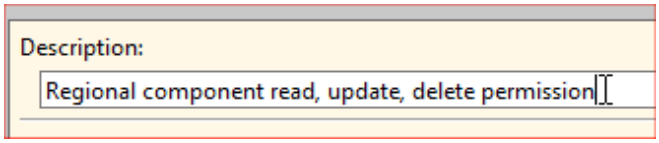
2 To start from an existing permission, select the rule you wish to copy and click the **Copy** icon.



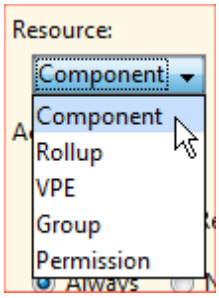
3 Whether you are Adding a new permission or starting from a Copy, aPriori prompts you enter a name, or to edit the default name created for the Copy. Click **OK** when done.



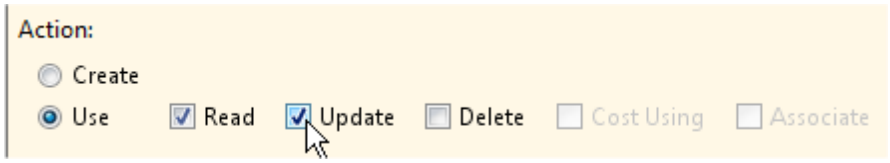
4 Enter a descriptive string in the Description field.



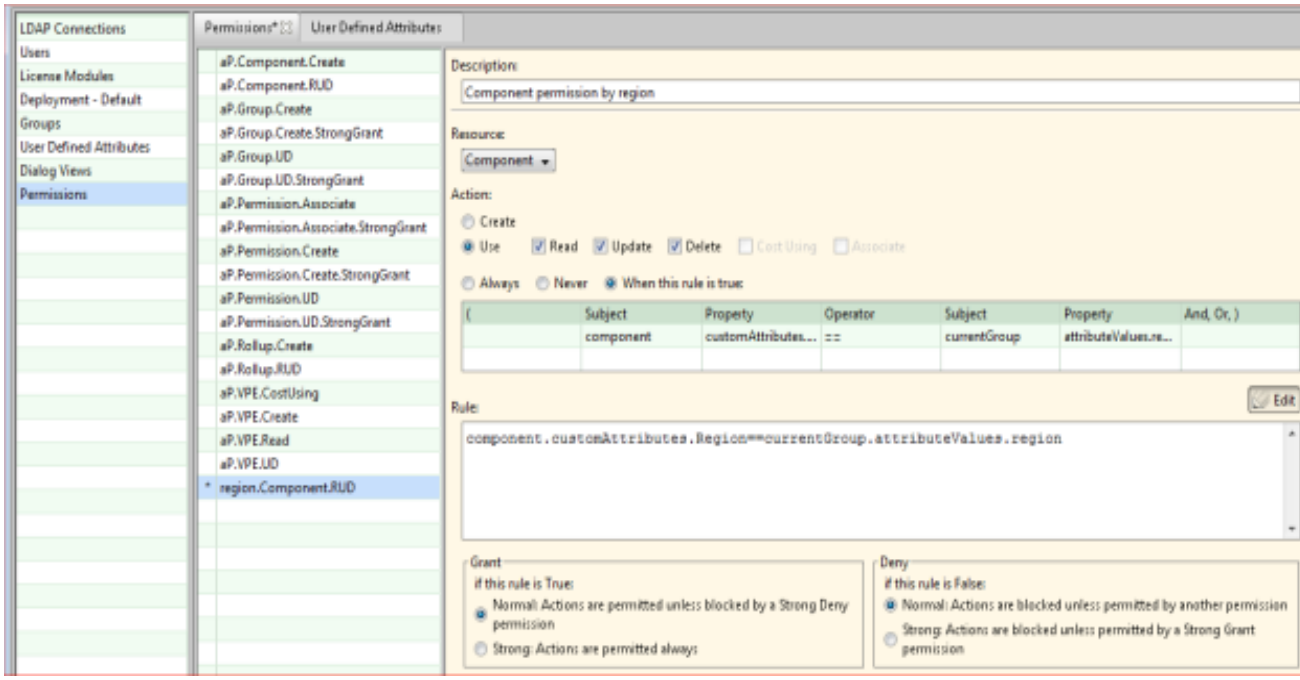
5 From the **Resource** drop-down menu, select the target of this permission:



6 From the Action section, select either **Create** or **Use**. If you select **Use**, you must select at least one of the actions provided. Only actions that are valid for your selected resource are enabled.

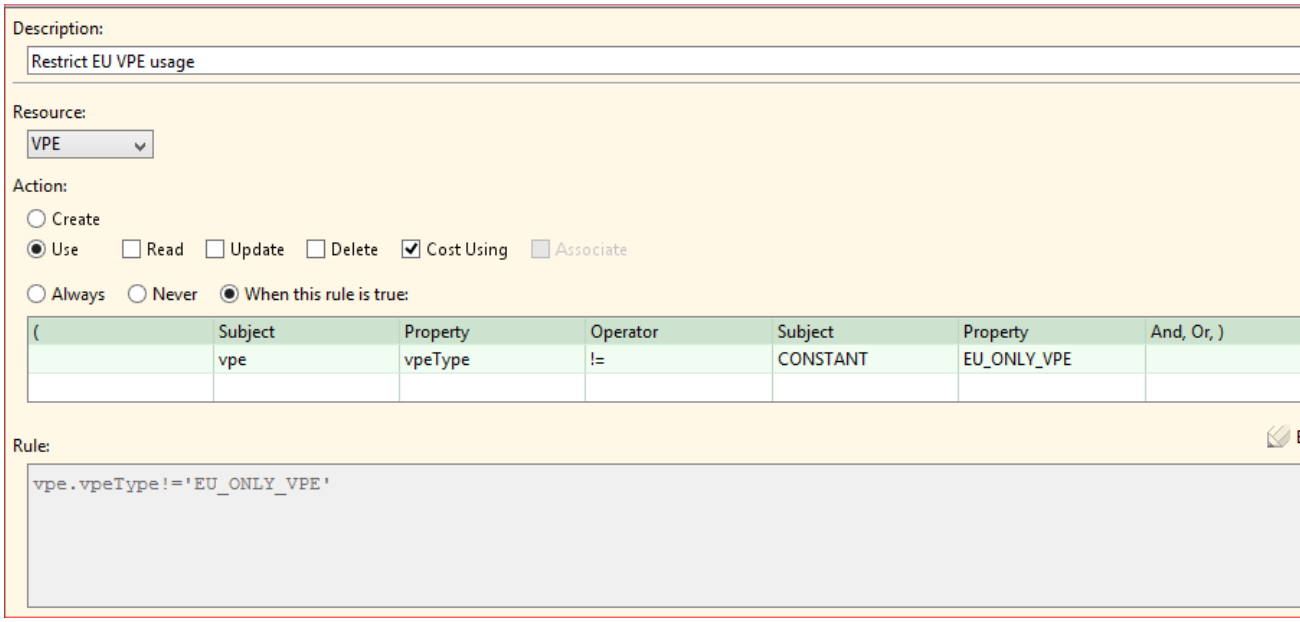


7 Set the rule for this permission. If the permission is to be **Always** or **Never** true, aPriori automatically sets the permission and displays it in the Rule field. You cannot edit an Always or Never rule. However, if you want to define a more complex rule that, for example, compares the value of a group attribute with the value of a scenario User Defined Attribute, you can select **When this rule is true** and build up the expression. The rule is displayed in the Rule field, and you can click the pencil **Edit** icon to edit and fine-tune the rule. Do not forget to click the pencil **Edit** icon again when leaving the Rule field.



Note that you build rules in a general left-to-right workflow: and your available options are determined by the settings that you choose for the **Resource:** pull-down menu and the Action: buttons and checkboxes:

- Choose the “noun” for the rule from the **Subject** column drop-down menu. For example, “vpe”.
- Choose the attribute for the Subject from the **Property** drop-down menu. For example, “vpeType”.
- Choose a comparison symbol from the **Operator** drop-down menu. For example, “!=” for “not equal”.
- Choose another noun for the right side of the equation from the second **Subject** drop-down menu. For example, “CONSTANT” to specify a string,
- Choose another attribute for the right side of the equation. Note that if the drop-down menus do not provide the value you need, you can always click the pencil icon from the upper-right of the **Rule:** field and manually edit the rule that you are building. For example, the only canned values available from the **Property** drop-down menu when the **Subject** is “CONSTANT” are “false: and “true”. But maybe you want to test that the value of the “vpeType” is not a label that your company has created and assigned to European VPEs, such as “EU_ONLY_VPE”. In this case you can manually edit the Rule field and enter ‘EU_ONLY_VPE’ (in single quotes) on the right side of the “!=” operator, You could then assign this rule to groups of North American users to ensure that they cannot use the EU VPE(s).



Group Membership Considerations

This section describes permissions added in Release 2017 R1.

To create a Group Membership permission

As of Release 2017 R1, a new a new Resource target (“User”) and a related new Action (“Member Of”) were introduced to support automatic group population for any group that has a membership setting of “Automated” (see *Group Membership settings*).

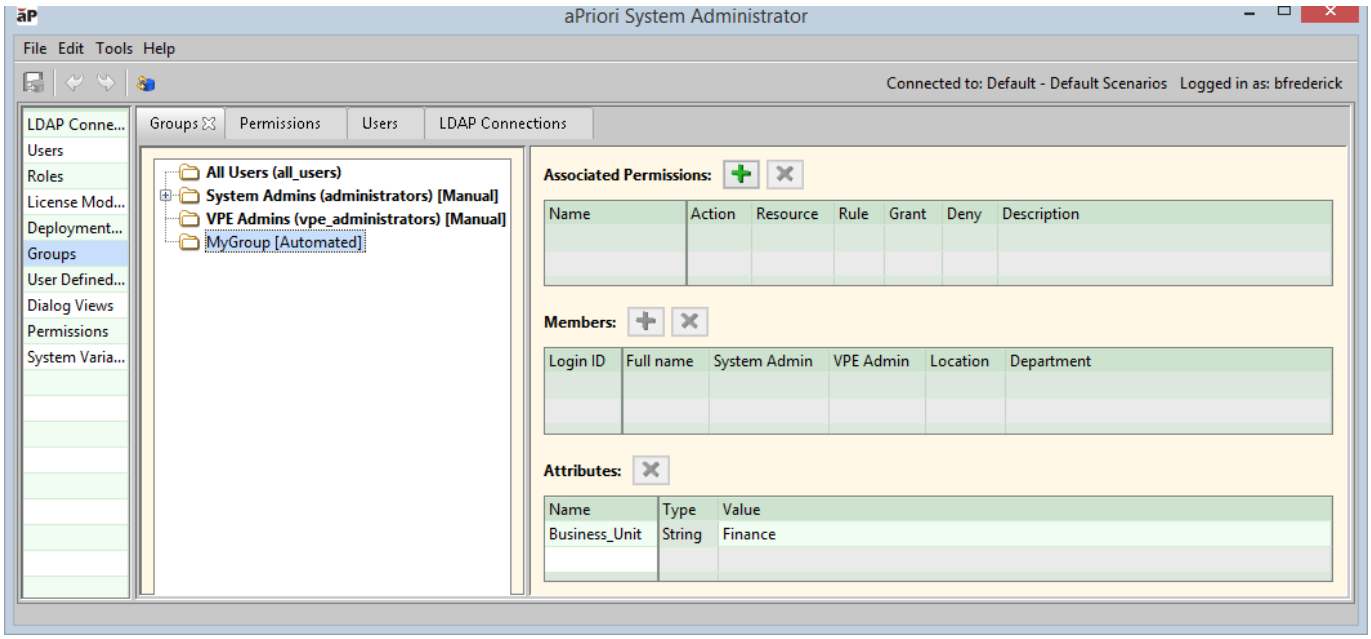
These permissions will typically compare attributes of the user to the attributes of the group. For example,

```
User.department == currentGroup.attributeValues.AC_Department
```

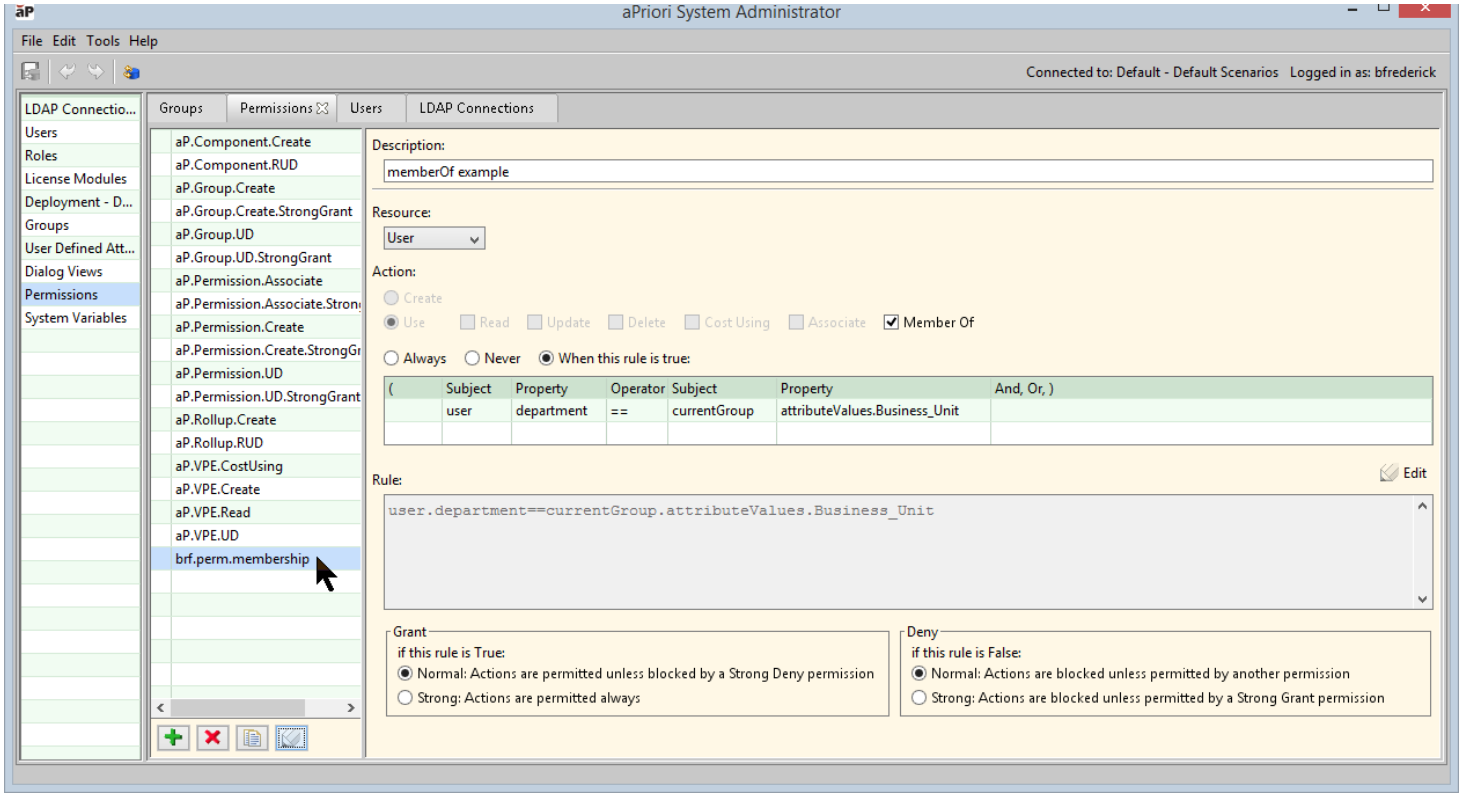
When the group membership permission is checked, it denies or grants to indicate whether the user is a member of the current group. A group can have more than one membership permission.

Here is a simple example that tests whether the value in the user attribute “department” is equal to the group’s attribute “Business_Unit”.

1 In the **Groups** tab, create a group with a Group Membership setting of **Automated**, and create an attribute called “Business_Unit” of type **String** with a value of “Finance”.



2 Go to the **Permissions** tab and create a permission with a **User** resource and **Member Of** action that compares the user's department to the group's Business Unit:

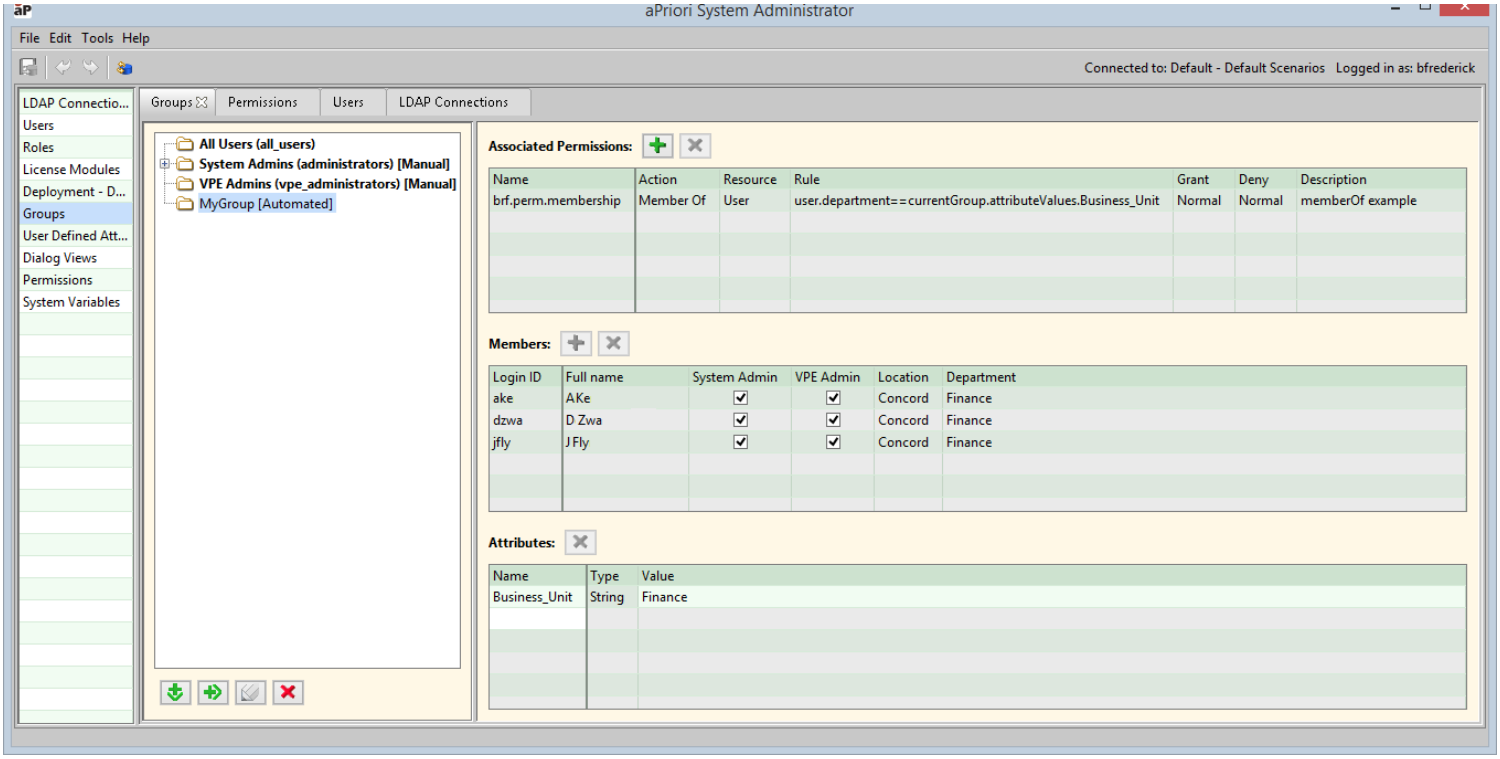


3 **Publish** when done.

4 Return to the **Groups** tab and add the permission that you just created. Click the **Add** (+) icon next to **Associated Permissions**, search for the name of the permission you just created, then select it and click **OK**.

5 **Publish** when done.

Your members list should become populated with all users who match the query:



Group Membership Algorithm:

Here is how the Group Membership Process handles these permissions:

1 For each Automated access control group, check if a user is a member of that group by invoking each group membership permission.

2 Apply normal access control semantics if there is more than one of these permissions on the group (for example, a strong deny would trump a normal grant). In addition, since a user would also be a member of the group's parent groups (all the way up to All Users), all group membership permissions for each group in this branch are checked.

> If a user is a member of a subgroup, then they are also a member of the parent group. This access control rule cannot be violated. So, if a parent group is also an Automated group and has a group membership permission that returns a deny, the user would be a member of both the child and parent groups. If the parent group's permission returned a strong deny then the user would not be a member of either group.

3 Overwrite the group membership for all AC groups that are tagged "Automated".

"None" groups are also affected (for example if a None group was a parent of an Automated group.

"Manual" groups can also be affected (for example if a parent group is an automated group and its group membership is changed the manual group could be affected)

The interaction between group types and parent/child groups is summarized below:

Parent/Child	Manual	Automated	None
Manual	Child users are in the Parent Group (If removed from the Parent Group they are removed from the child group too, but not vice versa).	Child users are included in the Parent Group (cannot be removed from the Parent Group).	Indirect child users are members in the Parent Group.
Automated	Child users have to pass Parent's membership rules to be allowed into the Child group. For example, a normal deny by the parent would allow manual entry into the Child group. A Strong Deny would not allow entry into the Child group. (Manual membership acts like a grant.)	Child users have to pass both Child and Parent's membership rules to be allowed into the Child group. For example, a normal deny by the parent and grant by the child would allow entry into the Child group.	Indirect child users are members in the Parent Group.
None	Child users are in the Parent group.	Child Users have to pass Child membership rules and then they are members in the Parent group.	Indirect child users are members in the Parent Group.

Note: This table focuses on one level of the hierarchy; but these rules are checked for the full branch of the tree (from the leaf nodes right up to All Users).

For example:

1 Parent = Automated, Child = Manual or Automated

2 User X is added to a Child group which satisfies the Parent membership rule

3 At some future time, the Parent membership rule is changed (the Permission is changed, or one of the attributes that the Permission uses changes) such that User X no longer satisfies the Parent membership rule

Membership of the Child group depends on what the Parent membership rule does:

- If the parent rule is a normal deny, and the child membership rule returns grant then nothing would change (since we treat a manual association as a grant in the case that the child group is manual).
- If the parent rule is a strong deny then the parent membership permission wins/trumps and the user is removed from the child group (and all the way down the chain unless there was a strong grant).

Using the Access Control Command Line

As of Release 2018 R3 SP1, you can perform Access Control group management tasks from the command line, using a .csv spreadsheet approach similar to the aPriori BOM Loader and Bulk Costing functionality.

This feature supports the following actions:

- **ADD –** Create one or more new groups, specifying the parent group path, group type, user defined attributes, and/or associated permissions.
- **REMOVE –** Delete one or more existing groups.
- **MODIFY –** Update one or more groups, specifying a new name, changed attributes, and/or changed associated permissions.

The command to run this functionality is:

```
<apriori_install>\bin\groupLoad.cmd
```

```
Once the command completes, group membership is recalculated to reflect
the changes.
```

groupLoad.cmd syntax

```
groupLoad.cmd <importFilePath> {<userName>} {<userPassword>}
```

where:

<importFilePath> is the full path to the import spreadsheet file

<userName> is the aPriori user account under which the changes will be made.

<userPassword> is the password for the aPriori user.

<userName> and *<userPassword>* are optional if Single Sign On is enabled. The group loader will try to log in using the current Windows user login if they are omitted. Note that the user should be a member of the Super Users group.

Spreadsheet Template

A template spreadsheet is available at:

```
<apriori_install>\ext\group-loader-plugin\templates\
   groupCreationTemplate.xls
```

The template has four sheets:

- groups
- attributes
- permissions
- sheetIndex

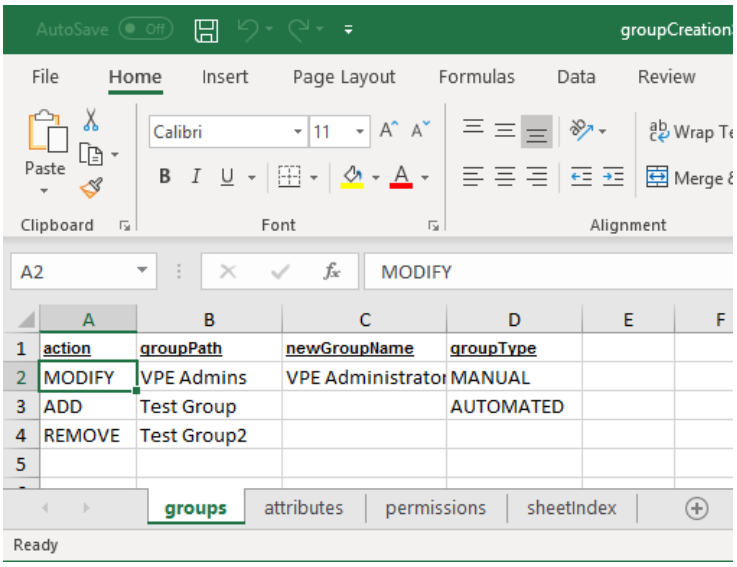
Groups sheet

The groups sheet has the following columns:

- **action –** contains one of three values (must be in upper-case):
  - **ADD**
  - **MODIFY**
  - **REMOVE**
- **groupPath –** contains the path where the group being modified or deleted can be found. In case of new group creation - this will be the path of the new added group. If group already exist - then import will do nothing (i.e. there will be no duplicates after import).
- **newGroupName –** contains new group name in case of group modification (applicable only for MODIFY action). If value in this column is not set, then group name will not be changed.
- **groupType –** contains one of three values (in upper case):
  - **NONE**
  - **MANUAL**
  - **AUTOMATED**

  The types will be applied as follows:
  - If action=ADD, this value will be the group membership type of the newly created group.
  - If action=MODIFY, this value will be the new group membership type (if desired). If this value is not set, the membership type of the group will not be changed.
  - If action=REMOVE, this value is not needed: the group to be removed is searched only by its path.

Example:



Attributes Sheet

The attributes sheet has the following columns:

- **action –** contains one of three values (must be in upper-case):
  - **ADD –** create a new attribute
  - **MODIFY –** edit an existing attribute
  - **REMOVE –** delete an existing attribute
- **groupPath –** contains the path of the group to which the attribute belongs.
- **attributeName –** contains the name of the attribute which should be modified or removed. If action=ADD, this value will be the name of the newly created attribute. If the attribute already exists, import will do nothing (i.e., duplicates will not be created).
- **attributeType –** contains one of three values:
  - **Boolean**
  - **String**
  - **List**
  - **Double**

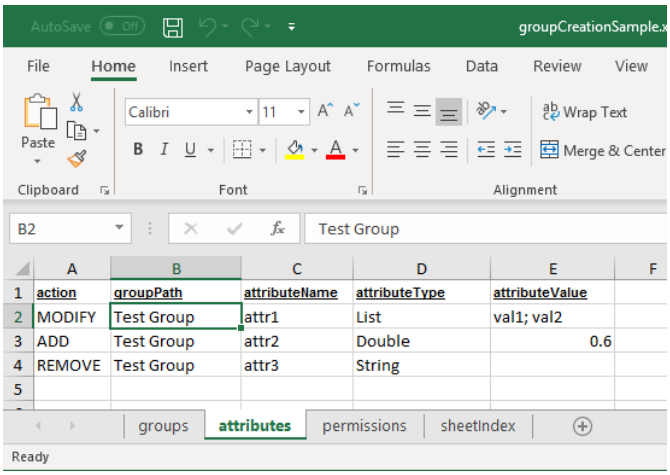
  If action=ADD, this value will be the type of newly created attribute.

  If action=MODIFY, this value should contain the exact type of attribute to be found and modified. **Note:** The attribute type itself cannot be modified.

  If action=REMOVE, this value it not needed; the attribute to be removed will be searched by group path and name.
- **attributeValue –** contains the value of the newly created attribute (when action= ADD), or the new value of the modified attribute (when action=MODIFY).

  Double values should contain decimal points. List values should be separated with semicolons.

Example:



	A	B	C	D	E	F
1	action	groupPath	attributeName	attributeType	attributeValue	
2	MODIFY	Test Group	attr1	List	val1; val2	
3	ADD	Test Group	attr2	Double	0.6	
4	REMOVE	Test Group	attr3	String		
5						

Permissions Sheet

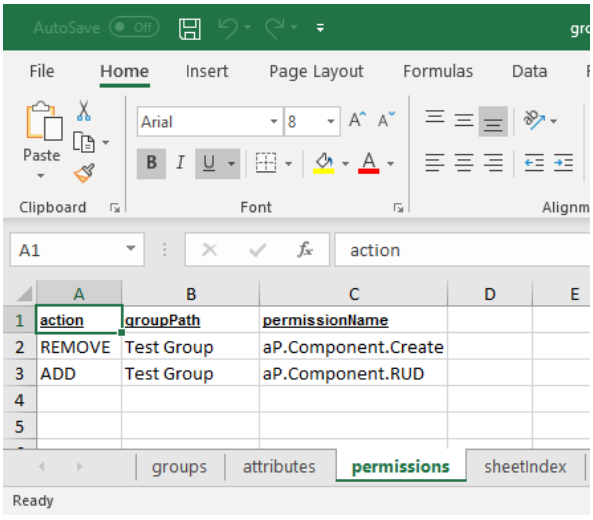
The permissions sheet has the following columns:

- **action** – contains one of two values (must be in upper-case):
  - ADD – create a new group permission.
  - REMOVE – delete a group permission.
- **groupPath** – contains the path of the group to which the permission belongs.
- **permissionName –** contains the name of the permission.

  If action=ADD, this value will be the name of the newly added permission. If the permission has been added already, import will do nothing (i.e. duplicates will not be created).

  If action=REMOVE, this value is the name of the permission to be removed.

Example:



	A	B	C	D	E
1	action	groupPath	permissionName		
2	REMOVE	Test Group	aP.Component.Create		
3	ADD	Test Group	aP.Component.RUD		
4					
5					

SheetIndex Sheet

An Excel index sheet for navigating to other sheets. DO NOT EDIT.

Import Results and Error Messages

Upon completion, the group loader displays a short summary:

```
groupLoad:
         [java] ------------------------------------------------
         [java]                 BEGIN GROUP LOAD
         [java] ------------------------------------------------
         [java]
         [java]
         [java] Import summary:
         [java]
         [java] Groups added: 1
         [java] Groups modified: 0
         [java] Groups removed: 0
         [java] Groups processed with errors: 2
         [java]
         [java] Permissions added: 0
         [java] Permissions removed: 0
         [java] Permissions processed with errors: 2
         [java]
         [java] Attributes added: 1
         [java] Attributes modified: 0
         [java] Attributes removed: 0
         [java] Attributes processed with errors: 2
         [java]
         [java] See apriori.log for details.
         [java]
         [java] ------------------------------------------------
         [java]                 END GROUP LOAD
         [java] ------------------------------------------------
         [java]

BUILD SUCCESSFUL

Total time: 10 seconds
```

If errors are listed, check the apriori.log file, which is typically located in:

C:\<*userHomeDir*>\aPriori\<*aPrioriVersion*>\logs

Access Control of Access Control

This section covers the topic of how to apply Access Control to Access Control: the ability to control the access rights that your administrator(s) are granted for defining and modifying your organization's Access Control model. This is an advanced area and should be used only when you have a firm grasp of Access Control concepts.

For an example of how this functionality might be useful, consider a regulatory requirement that makes it necessary to restrict the access that some administrators have to certain data:

Due to export control regulations, only a small number of users in the USA region of a multi-national corporation should have access to the organization's export control information.

Administrators (and any other users) outside of the USA region should not be able to access this data.

These administrators should also not be able to create or modify the Access Control model to allow themselves or others to gain access to the data.

To configure this kind of Access Control model, aPriori provides a "Super Users" sub-group of the system-defined Administrators group (see Super Users). Administrators who belong to the Super Users group have extra privileges that other administrators do not have. Specifically, they are able to access any data in the aPriori system, and they have the ability to define and modify the Access Control model.

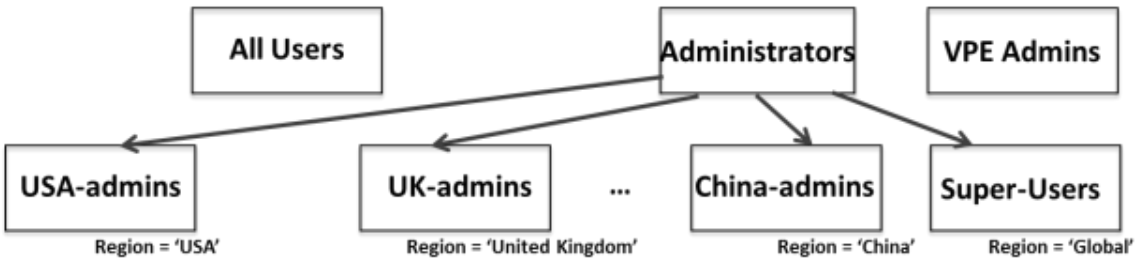
All administrators by default have full rights to access all data in the system. But Super User Administrators not only have these same permissions, they have these permissions with the "Strong Grant" option enabled. In addition, Super User Administrators also have the following key privilege:

> The unrestricted ability to create groups and assign permissions to groups.

You may recall that all users can have permissions that include components, roll-ups, and VPEs as resources. But Super User permissions can also target permissions and groups as resources. This is what gives Super Users their special capabilities.

Basic Rules for Access Control of Access Control

When creating sub-groups of the system-defined All Administrators group, mimic the structure that you have created for your users. For example, if you have created groups for your users based on regions such as USA, UK, and China, do the same for your administrators:



(The Super-Users group is automatically created by aPriori during installation.)

Only allow members of the Super Users group to CREATE, UPDATE, or DELETE permissions.

By default, only members of the Super-Users group have strongGrant permission to associate permissions. Other administrators can associate permissions, but only in their regions (this is controlled by Update permissions to the groups that represent regions).

Try to use parameterized permissions; do not use permission-specific rules by specifying the name of the permission. However, if you must use permission-specific rules then you should write an Associate (Add to Group) rule for every permission (i.e. that says who can associate it).

The following is a summary of the requirements for various aspects of Access Control of Access Control:

To create a Group or Permission: you need a CREATE permission defined on the Group and Permission resource types. (If you create a subgroup you will need an UPDATE permission on the parent group.)

To Delete a Group or Permission: you need a DELETE permission defined on the Group and Permission resource types.

To Update a Group: you need an UPDATE permission on the Group resource type.

Updating a group includes:

- Adding a permission to the group or removing it.
- Changing any attribute of the group (including its name)
- Adding or removing users

To Update a Permission: you need:

- UPDATE permission on the Permission itself
- UPDATE permission for any Group that the Permission is in

Updating a Permission includes updating any of its attributes (name, rule, resource, action, and so on).

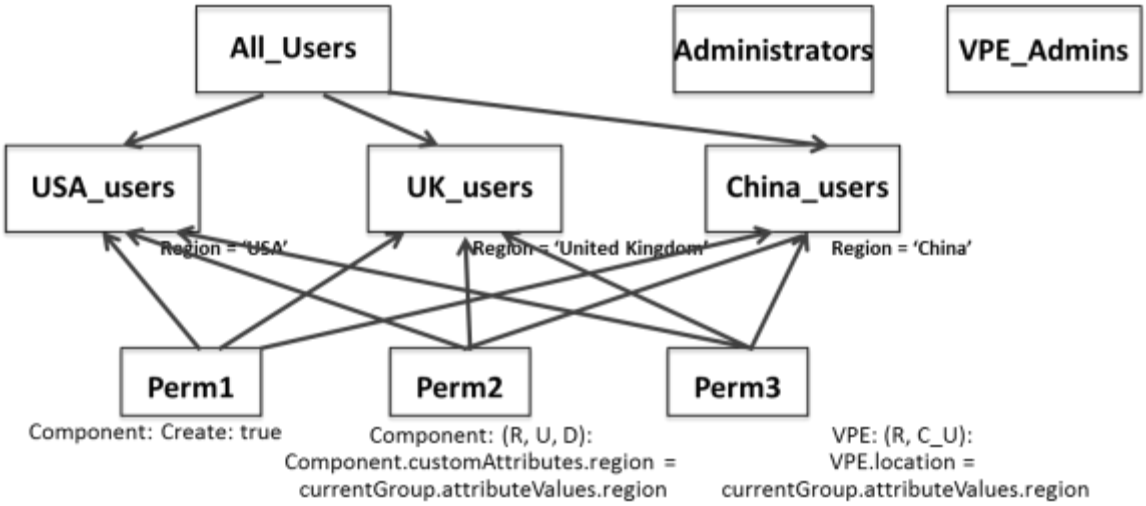
Notes:

- There are no user defined attributes on permissions.
- Rename requires UPDATE access on the permission and UPDATE access on all associated groups since it is an update to the permission.

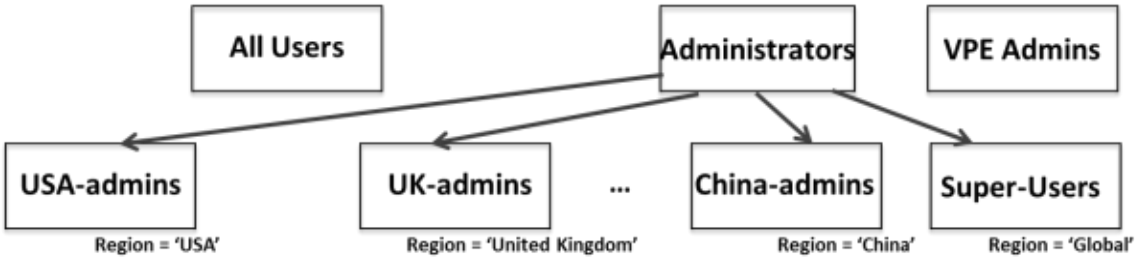
To Associate (Add) a Permission to a Group: you need ASSOCIATE permission on the Permission itself, and UPDATE permission on the Group.

Example Configuration

Assume that you already have groups configured for your users based on a region model, and you now wish to configure your administrators. Here is the existing user group configuration:



You should probably configure your administrator groups to reflect your user groups:



In this example, three region-based administrator groups have been created to reflect the three region-based user groups. The Super-Users group is created automatically by aPriori. Each of the region-based groups has an attribute called "Region" which is set to the same value used in the user groups (for example, "USA", "China", etc.). The Super-Users group's "Region" attribute is set to "Global".

Note: In an actual installation, you would also need to create VPE administrator groups. These might or might not also mirror the region-based groups, depending on the requirements for VPE creation, update, and delete.

Permissions for the different groups

All Users – By default, aPriori sets READ access to all Groups and all Permissions for all users:

- READ Permission true, Strong Grant
- READ Group true, Strong Grant

This is hard-coded and aPriori does not allow the creation of READ permissions on Groups and Permissions.

Super Users – By default the Super-Users group has the following permissions related to groups and permissions:

- aP.Group.Create.StrongGrant: CREATE Groups
- aP.Group.UD.StrongGrant: UPDATE and DELETE Groups

- aP.Permission.Associate.StrongGrant: ASSOCIATE Permissions ("Associate" means "Add Permission to Group")
- aP.Permission.Create.StrongGrant: CREATE Permissions
- aP.Permission.UD.StrongGrant: UPDATE and DELETE Permissions

Note that Super-Users should always have strongGrant permissions on all resource types, for all actions.

Administrators – In this scenario, members of the top-level Administrators group have the following permissions:

- CREATE, UPDATE, DELETE Permission, false (these are not denied, they are simply not explicitly granted)
- ASSOCIATE ("Add to Group") Permission, If permission.strongGrant == false (or permission.normalgrant == true)

Note: Non-Super User Administrators *cannot* associate "Strong Grant" permissions (or, put another way, they can only associate "Normal Grant" permissions.)

USA-Admins, UK-Admins, China-Admins – The regional administrators would have the following permissions:

UPDATE Groups [Normal Deny], currentGroup.attributeValues.region = Group.attributeValues.region

This rule ensures that admins can only update groups that are in their region – for example, Associate permissions to groups in their region.

Configuring Admin Roles Using Access Control Permissions

You can configure system and virtual-production-environment (VPE) administrator roles by using permissions to control whether the administrators can:

- Access none, some, or all the tabs in the System Admin Toolset and the VPE Toolset.
- Edit configuration settings in none, some, or all the accessible tabs within each toolset.

You can configure administrator access to these tools:

- System Admin tools:
  - System Administrator
  - Migration Import Tool
- VPE tools:
  - VPE Manager
  - Cost Model Workbench
  - Process Group Site Variables
  - Deployment Data

- BOM Loader

To configure an administrator role, you first create or modify permissions using one of these options from the **Resources** dropdown menu in the **Permissions** tab of the **System Administrator:**

- **System Admin**
- **VPE Toolset**

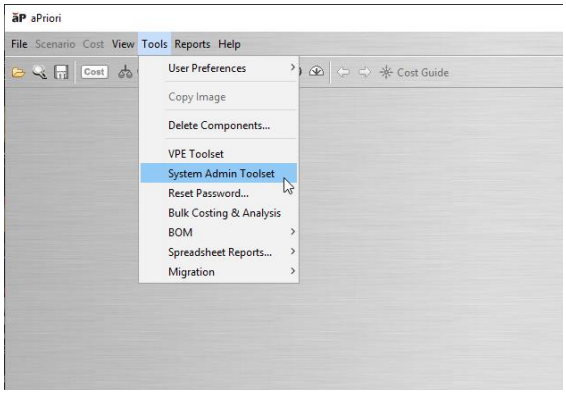
After you modify or create or the permission, to apply the permission to any user, you must associate the permission to a group that the member belongs to. Once you associate a permission to a group, the permission applies to all members of the group. If you want to apply a permission to a single user, create a group that contains only that user.

Create a Permission that Grants Access to the System Admin Toolset

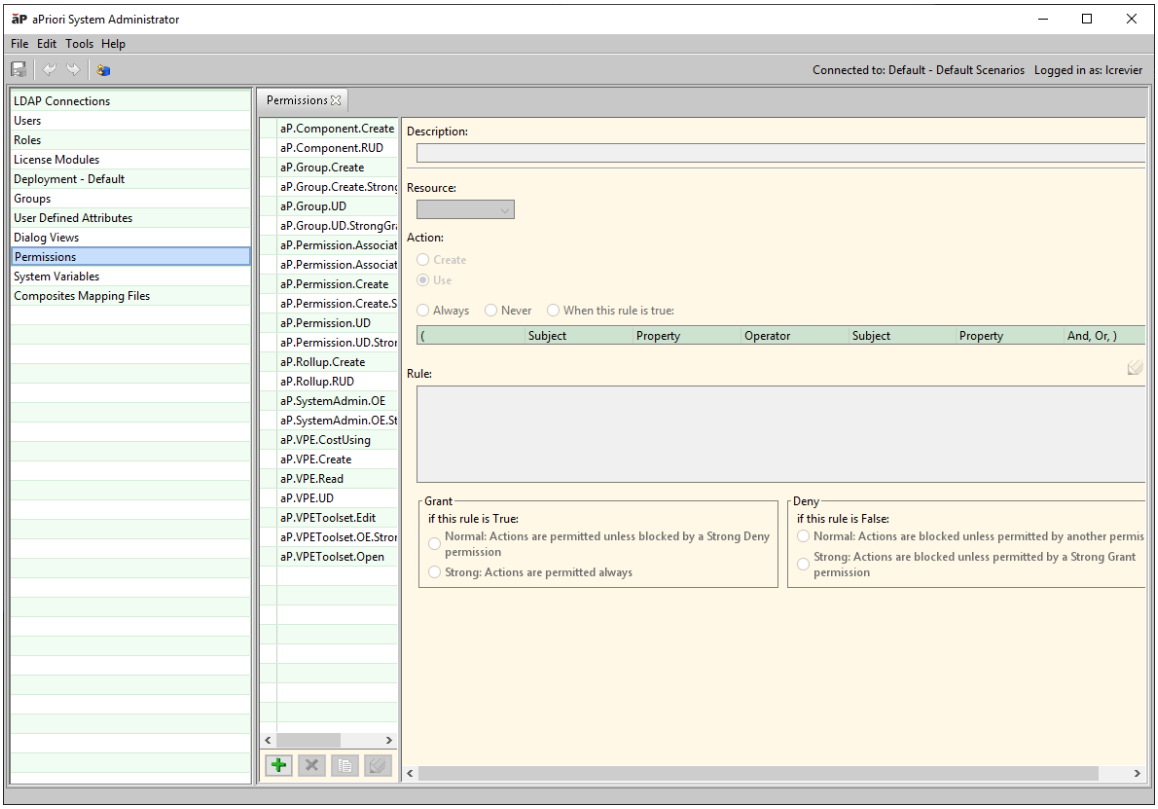
This example shows how to create a permission that:

- You can later associate to any groups
- Allow all members of the associated group to use all the tools in the System Administrator Toolset.

1. Open the System Admin Toolset. From the aPriori Professional menu bar, select **Tools > System Admin Toolset**.



2. Open the System Administrator. In the **System Admin Toolset** window, click the System Administrator,  , button.



3. Open the **Permissions** tab. In the **System Administrator** window, click **Permissions.**
4. In the **Permissions** tab, create and configure the new permission:

   4.1. Under the permissions list, select the add,  , button.

   4.2. **In the Input tab, for Permission Name** enter the name of the new permission. As a best practice, follow this naming convention *`YC.SystemAdmin.Grant`*, where *`YC`* represents the name of your company and then click OK.

   NOTE: Permission names that start with aP are aPriori defined permissions.

   4.3. In the Permissions tab, for:

   4.3.1. **Description** – Enter *`Grant access to all System Admin Toolset features`***.**

   4.3.2. **Resource** – From the dropdown menu, select **VPE Toolset**.

   4.3.3. **Action** – Select **Use, Open, Edit, and Always.**

   For this configuration, the generated rule is `true`.

   4.3.4. **Grant –** Select **Normal: Actions are permitted unless blocked by a Strong Deny permission.**

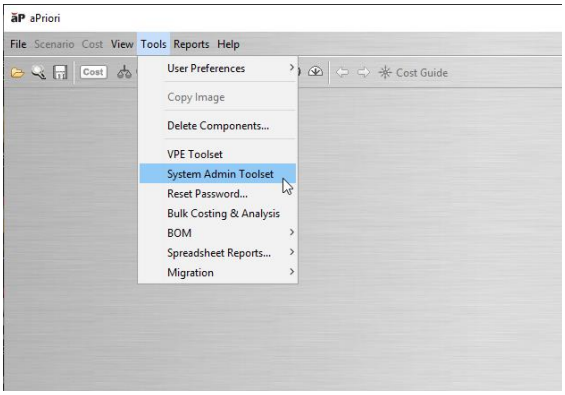
   4.3.5. **Deny –** Select **Normal: Actions are blocked unless permitted by another permission.**

5. To save your changes, click the Publish,  , button.

You can now apply the permission to all members of a group by associating the group to the permission. For an example that show you how to apply the permission, see Associate a Permission to a Group.

Associate a Permission to a Group

1. Open the System Admin Toolset. From the aPriori Professional menu bar, select **Tools > System Admin Toolset**.



2. Open the System Administrator. In the **System Admin Toolset** window, click the System Administrator,  , button.
3. In the **Groups** tab, in the groups list, select group directory for the group that you want to associate with a permission**.**
4. In **Associated Permissions,** select the Add Permissions to Group,  , button.
5. In the **Search Permissions** window, for **Name,** enter the name of the permission that you want to associate to the group and then click **Find.**
6. In the **Available Permissions** list, select the name of the permission that you want to associate to the group and then click **OK.**

Create a Permission that Grants Access VPE Admin Toolset

This example shows how to create a permission that:

- You can later associate to any groups
- Allow all members of the associated group to use all the tools in the VPE Toolset.

1. Open the **Permissions** tab. From the aPriori menu bar, select Tools **> System Admin Toolset > System Administrator > Permissions**.
2. In the **Permissions** tab, create and configure the new permission:
   2.1. Under the permissions list, select the add,  , button.
   2.2. In the Input tab, for Permission Name enter `YC.VPEToolset.Grant` and then click **OK.**
   2.3. In the Permissions tab, for:

2.3.1. **Description** – Enter *`Grant access to all VPE Toolset features`*.

2.3.2. **Resource** – From the dropdown menu, select **VPE Toolset**.

2.3.3. **Action** – Select **Use, Open, Edit,** and **Always.**

For this configuration, the generated rule is `true`

2.3.4. **Grant** – Select **Normal: Actions are permitted unless blocked by a Strong Deny permission.**

2.3.5. **Deny** – Select **Normal: Actions are blocked unless permitted by another permission.**

3. To save your changes, click the Publish,  , button.

You can now apply the permission to all members of a group by associating the group to the permission. For an example that show you how to apply the permission, see Associate a Permission to a Group.

For more information on creating or modifying permissions, see Permissions Tab.

Grant Permissions Selectively by Using `uiElementValue` Properties

You can selectively provide access to tools and features in the System Admin and VPE Toolsets by configuring tab access. To configure tab access, use the `uiElement` and `uiElementValue` subjects and properties. The `uiElement` subject has only one possible property, `name`. The `uiElementValue` subject has several properties. Each `uiElementValue` property maps to a first- or second-level tab in the System Admin or VPE Toolset.

Each tool in the System Admin and VPE Toolsets contains at least one tab. Tabs contain settings that you use to configure the tools. Each tab has a name that differs from the names of all other tabs for that tool. Tabs are classified as either first- or second-level tabs. If a tool has only one first-level tab, the name of the tool and tab are identical. For example, the single, first-level Migration Import Tool tab is named Migration Import Tool and the single, first-level VPE Manager tab is named VPE Manager. For these cases, when you open the tool, the first-level tab appears.

If a tool has more than one first-level tab, when you open the tab, the first-level tabs appear in clickable a list.

A second-level tab is a tab that appears or is listed on a first-level tab. First- and second-level tab sets are said to be multi-generational. The first-level tab is referred to as a parent tab, while each second-level tab is referred to as a child tab.

For example, in the System Admin Toolset, the System Administrator tool contains a first-level tab named Groups. The **Groups** tab contains three second-level tabs:

- **Associated Permissions**
- **Members**
- **Attributes**

Each of the three second-level tabs is a child of **Groups**, which is their first-level parent tab.

Similarly, the **Data Deployment** tab is a first-level tab that you use to configure the Data Deployment tool in the **VPE Admin Toolset**. These second-level tabs are children of the **Data Deployment** tab:

- **Currency**
- **Process Groups**
- **Cost Taxonomy Display Names**

These tables include the `uiElementValue` property mappings for all the first- and second-level tabs in the System Admin and VPE Toolsets that you can control access to. Child tabs are listed in the indented cells of rows below the row that contains their parent tab.

Toolset	Tool	Tab Name	Level	`uiElementValue` Property
System Admin Toolset	System Administrator	LDAP connections	First	`ldapConnections`
		Users	First	`users`
		Roles	First	`roles`
		Licenses	First	`licenseModules`
		Deployments - Default	First	`deploymentDefault`
		Groups	First	`groups`
		Associated Permissions	Second	`groupPermissions`
		Members	Second	`groupMembers`
		Attributes	Second	`groupAttributes`
		User Defined Attributes	First	`userDefinedAttributes`
		Dialog Views	First	`dialogViews`
		Permissions	First	`permissions`
		System Variables	First	`systemVariables`
		Composites Mapping Files	First	`compositeMappingFiles`
	Migration Import Tool	Migration Import Tool	First	`migration`

NOTE: Granting “Open” permission for the Migration Import Tool, enables users to run the utility but does not enable users to edit the utility settings. Users can edit the utility settings only if you grant them “Edit” permission for the Migration Import Tool.

Toolset	Tool	Tab Name	Level	`uiElementValue` Property
VPE Toolset	VPE Manager	VPE Manager	First	`vpeManager`
	Cost Model Workbench	Cost Model Workbench	First	`costModelEditor`
	Process Group Site Variables	Process Group Site Variables	First	`procesGroupSiteVariables`
	Deployment Data	Deployment Data	First	`deploymentData`
		Currency	Second	`deploymentDataCurrency`
		Process Groups	Second	`deploymentDataProcessGroups`
		Cost Taxonomy Display Names	Second	`deploymentDataCostTaxonomyDisplayNames`
	BOM Loader	BOM Loader	First	`bomLoader`

NOTE: Granting "Open" permission for the BOM Loader, enables users to run the utility but does not enable users to edit the utility settings. Users can edit the utility settings only if you grant them "Edit" permission for the BOM Loader.

All the open and edit permissions that you apply for the **vpeManager** settings are extended to these plugins:

- Push Plant Variables to Descendants
- Add/Update Machine Field
- Add/Update Material Field
- Create New Process
- Search CSL

To access the plugins, in the VPE Toolset menu bar, click **Tools.**

Grant Admin Permissions to Only Users in a Certain Group

This example shows how you can grant administration permissions to only the users in a certain group by using `uiElementValue` properties. To enable open and edit rights to the System Admin Toolset for managing users only, the example applies the **users** `uiElementValue` property:

1. Open the **Permissions** tab. From the aPriori Professional menu bar, select **Tools > System Admin Toolset > System Administrator > Permissions.**
2. In the **Permissions** tab, create and configure the new permission:

   2.1. Under the permissions list, select the Add,  , button.

2.2. In the Input tab, for **Permission Name** enter `YC.SystemAdmin.Users.OE` and then click **OK.**

2.3. In the **Permissions** tab, for:

2.3.1. **Description** – Enter `Allow configured admins rights to the manage users.`

2.3.2. **Resource** – From the dropdown menu, select **System Admin**.

2.3.3. **Action** – Select **Use, Open, Edit**, and **When this rule is true. For:**

2.3.3.1. **Subject** – Select `uiElement`**.**

2.3.3.2. **Property** – `name` **is the only option.**

2.3.3.3. **Operator** – Select `==`**.**

2.3.3.4. **Subject** – Select **uiElementValue.**

2.3.3.5. **Property** – Select **users.**

For this configuration, the generated rule is `uiElement.name==uiElementValue.users`

2.3.4. **Grant –** Select **Normal: Actions are permitted unless blocked by a Strong Deny permission.**

2.3.5. **Deny –** Select **Normal: Actions are blocked unless permitted by another permission.**

3. To save your changes, click the **publish,**  **, button.**

Multi-Generational Access Control Permission Interactions

When you limit the open or edit rights for multi-generational System Admin and VPE Toolset tabs, there are permission interactions. This table shows how the open and edit rights that you grant/deny for first-level tabs and their second-level children tabs interact to affect what users can see and edit.

Tab Permissions				Admin Rights Interaction Effects
First-Level (Parent) Tab		Second-Level (Child) Tab		
Open	Edit	Open	Edit	
X	X	X	X	Can see and edit both the parent and the parent/child relationship.
X	X	X	-	Can see both the parent and child. Can edit the parent but cannot edit the child.
X	X	-	X	Can see and edit both the parent and the parent/child relationship
X	-	X	X	Can see both the parent and child but cannot edit the parent or the parent/child relationship.
-	X	X	X	Can see and edit both the parent and the parent/child relationship.

For example, this table shows the interaction effects for the **Groups** and **Members** tabs, which are multi-generational tabs in the System Admin Toolset System Administrator tool.

Tab Permission				Admin Rights
Groups		Members		
Open	Edit	Open	Edit	
X	X	X	X	Can: • See groups. • Edit groups. • See user membership for each group. • Edit user membership for each group.
X	X	X	-	Can: • See groups. • Edit groups. • See user membership for each group. Cannot edit user membership for each group.
X	X	-	X	Can: • See groups. • Edit groups. • See user membership for each group. • Edit user membership for each group.
X	-	X	X	Can: • See groups. • See user membership for each group. Cannot: • Edit groups. • Edit user membership for each group.
-	X	X	X	Cannot access group or membership data at all, that is, cannot: • See groups. • Edit groups. • See user membership for each group. • Edit user membership for each group.

Likewise, this table shows the interaction effects for the **Deployment Data** and **Currency** tabs, which are multi-generational tabs in the VPE Toolset Deployment Data tool.

Tab Permission				Admin Rights
Deployment Data		Currency		
Open	Edit	Open	Edit	
X	X	X	X	Can: • See deployment options. • Edit deployment options. • See currency for each deployment option. • Edit currency for each deployment option.

Tab Permission				Admin Rights
Deployment Data		Currency		
Open	Edit	Open	Edit	
X	X	X	-	Can: • See deployment options. • Edit deployment options. • See currency for each deployment option. Cannot edit currency for each deployment option.
X	X	-	X	Can: • See deployment options. • Edit deployment options. • See currency for each deployment option. • Edit currency for each deployment option.
X	-	X	X	Can: • See deployment options. • See currency for each deployment option. Cannot: • Edit deployment options. • Edit currency for each deployment option.
-	X	X	X	Cannot access deployment or currency data at all, that is, cannot: • See deployment options. • Edit deployment options. • See currency for each deployment option. • Edit currency for each deployment option.

aPriori Professional Out-of-Box (OOB) Access Control Model

When you first install aPriori Professional, no system admin, VPE admin, or super-user group, and therefore, no user in those groups, is denied the right to open and edit any data or to access and configure any associated tools. It is not that fresh installation does not include an Access Control model for these groups. Rather, it is aPriori Professional ships with an Out-of-Box (OOB) Access Control Model that includes a set of permissions that is designed to make it appear as if there is no access control for these groups.

In aPriori Professional 2019 R2, access control is extended to allow admins to grant or deny access and editing rights to tools and features in the System Admin Toolset and VPE Toolset. The OOB Access Control model is updated accordingly so that the default behavior for aPriori Pro is preserved. That is, when you first install aPriori Professional, no system admin, VPE admin, or super-user group, and therefore, no user in those groups, is denied access to the tools or features in any associated toolsets.

Note that because the OOB Access Control Model denies absolutely no permissions, it is not a good starting point for implementing a custom Access Control model.

The Root Access Control Model is a better starting point for implementing a custom Access Control Model because it includes a set of permissions that provide a “best practices” Access Control baseline model. At your request, the aPriori Services team

can deploy the Root Access Control Model and add extensions to meet the specific requirements of your organization. For information on the aPriori Professional Root Access Control Model, see

In the Groups tab, in the groups list, select group directory for the group that you want to associate with a permission**.**

3.1. In **Associated Permissions,** select the Add Permissions to Group,  , button.

3.2. In the **Search Permissions** window, for **Name,** enter the name of the permission that you want to associate to the group and then click **Find.**

3.3. In the **Available Permissions** list, select the name of the permission that you want to associate to the group and then click **OK.**

Root Access Control Model for aPriori Professional .

As of aPriori release 2019 R2, these OOB permissions and associations are required:

- aP.VPEToolset.Open – Grants VPE toolset access to All Users.
- aP.VPEToolset.Edit – Grants VPE toolset access to VPE Admins.
- aP.SystemAdmin.OE – Grants System Administrator toolset access to System Admins.
- aP.SystemAdmin.OE.StrongGrant – Grants System Administrator toolset access to Super Users
- aP.VPEToolSet.OE.StrongGrant – Grants VPE toolset access to Super Users

Migrate or Import to the Current Release from a Release Prior to 2019 R2.

If you are migrating or exporting an access control model from a release that is earlier than 2019 R2 to the current release, you must manually add the required OOB permissions for the current release. For example, to create and associate required OOB permissions for the current release, as indicated in the table:

Required Permission Name	Resource	Actions	Grant	Deny	Associated Group
aP.VPEToolset.Open	VPE Toolset	• Use • Open • Always	Normal	Normal	All Users
aP.VPEToolset.Edit	VPE Toolset	• Use • Edit • Always	Normal	Normal	VPE Admins

Required Permission Name	Resource	Actions	Grant	Deny	Associated Group
aP.SystemAdmin.OE	System Admin	• Use • Open • Edit • Always	Normal	Normal	System Admins
aP.SystemAdmin.OE. StrongGrant	System Admin	• Use • Open • Edit • Always	Strong	Normal	Super Users
aP.VPEToolSet.OE. StrongGrant	VPE Toolset	• Use • Open • Edit • Always	Strong	Normal	Super Users

1. Log in as a super user
2. Open the **Permissions** tab. From the aPriori Professional menu bar, select **Tools > System Admin Toolset > System Administrator > Permissions.**
3. Create the required permissions. For each permission:
   3.1. In the **Permissions** tab, create and configure the new permission:
   3.2. Under the permissions list, select the Add,  , button.
   3.3. In the Input tab, for **Permission Name** enter the permission name, as indicated by the second column of the table, and then click **OK.**
   3.4. In the **Permissions** tab, for:
      3.4.1. **Description** – Reenter the name of the permission. For example, for the aP.VPEToolset.Open permission, enter `aP.VPEToolset.Open` for the description.
      3.4.2. **Resource** – From the dropdown menu, select the value indicated by the **Resource** column of the table.
      3.4.3. **Action** – Select the actions indicated by the **Action** column of the table. Then, for:
         3.4.3.1. **Subject** – Select `CONSTANT`.
         3.4.3.2. **Property** – Select `true`.
         For every one of these required permissions, the generated rule is `true`.
      3.4.4. **Grant –** Select the value that by indicated in the **Grant** column in the table.
      3.4.5. **Deny –** Select the value that is indicated by the **Deny** column in the table.

4. To save your changes, click the publish,  , button
5. Associate the new permissions to the appropriate groups, as indicated by the name of the group in the last column of the table.
   5.1. In the **Groups** tab, in the groups list, select group directory for the group that you want to associate with a permission**.**
   5.2. In **Associated Permissions,** select the Add Permissions to Group,  , button.
   5.3. In the **Search Permissions** window, for **Name,** enter the name of the permission that you want to associate to the group and then click **Find.**
   5.4. In the **Available Permissions** list, select the name of the permission that you want to associate to the group and then click **OK.**

Root Access Control Model for aPriori Professional

The Root Access Control Model includes a set of permissions that provide a “best practices” baseline configuration for implementing a custom Access Control Model. At your request, the aPriori Services team can deploy the Root Access Control Model and add extensions to meet the specific requirements of your organization.

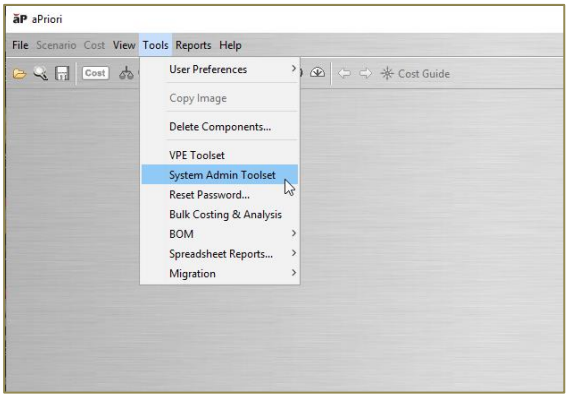
Access the Root Access Control Model Permissions

Prerequisite: Before you can access the Root Access Control Model, you must import a file that you can attain from the aPriori services organization.

NOTE: if you access the Root Access Control Model in the production environment every user becomes immediately blocked. You should only access the Root Access Control Model permissions in a development system.

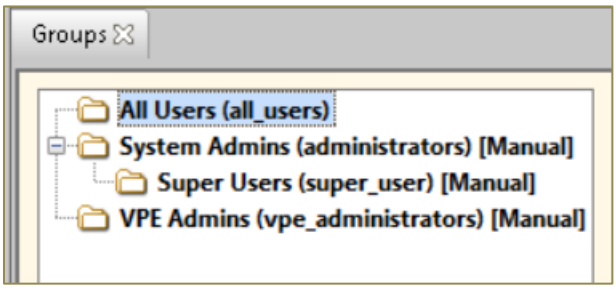
To access the Root Access Control Model permissions, open the **Groups** tab:

1. Open the System Admin Toolset. From the aPriori Professional menu bar, select **Tools > System Admin Toolset**.



2. Open the System Administrator. In the **System Admin Toolset** window, click the System Administrator,  , button.
3. Open the **Groups** tab. In the System Administrator window, click **Groups**.

4. To see the **Super Users (super_user)** folder, expand the **System Admins (administrators)** folder.



5. To examine the **Associated Permissions, Members,** and **Attributes** for each group, select the relevant group folder.

The root model enforces these rules for members of each of these OOB groups:

- **All Users** have almost no permissions by design. Rather than removing a long list of permissions that you want to deny, this construct allows you to add the typically shorter list of permissions that you want to grant.
- **Super Users**:
  - Have full access to data, VPEs, and the AC model
  - Cannot be blocked by Access Control permissions that restrict access to the System Admin Toolset or to the VPE Toolset.
- **System Admins** have:
  - Full access to end user data and to the System Admin tools
  - Limited access the Access Control Model
  - No access to VPEs. For VPE access, System Admins can be added to a VPE group.
- **VPE Admins** have:
  - Full access to VPEs (CRUD and Cost Using)
  - No access to end user data
  - Full access to the VPE toolset.

The aPriori Professional 2019 R2 Root Access Control Model is built upon the Root Access Control Model for the prior release. Differences between the base model for the 2019 R2 and the base model for the prior release are:

- System Admins and VPE Admins are enabled with full access to their respective tools.
- A new System Admins group permission, ap.SystemAdmin.OE, enables all the UI features in the System Admin tool.
- A new VPE Admins permission, aP.VPE.Toolset.OE, which opens all the UI features in the VPE Toolset.
- Two new Super User permissions:
  - aP.SystemAdmin.OE.StrongGrant

- aP.VPEToolset.OE StrongGrant

The tables show how the aPriori Professional 2019 R2 Root Access Control Model defines permissions for OOB groups.

All Users Group Permissions

Name	Action	Resource	Rule	Grant	Deny
aP.Component.Create	Create	Component	true	Normal	Normal
aP.Rollup.Create	Create	Rollup	true	Normal	Normal

System Admins Group Permissions

Name	Action	Resource	Rule	Grant	Deny
aP.Component.RUD	• Delete • Read • Update	Component	true	Normal	Normal
aP.Group.Create	Create	Group	true	Normal	Normal
aP.Group.UD	• Delete • Read	Group	Group.name! = 'super_user'	Normal	Normal
aP.Permission.Associate.OnlyNormal	Associate	Permission	Permission.normalGrant == true	Normal	Normal
aP.Rollup.RUD	• Delete • Read • Update	Permission	true	Normal	Normal
ap.SystemAdmin.OE	• Open • Edit	System Admin	true	Normal	Normal
aP.VPE.CostUsing	Cost Using	VPE	true	Normal	Normal
aP.VPE.Read	Read	VPE	true	Normal	Normal

VPE Admins Group Permissions

Name	Action	Resource	Rule	Grant	Deny
aP.VPE.CostUsing	Cost Using	VPE	true	Normal	Normal
aP.Rollup.Create	Create	VPE	true	Normal	Normal
aP.VPE.Read	Read	VPE	true	Normal	Normal

Name	Action	Resource	Rule	Grant	Deny
aP.VPE.UD	• Delete • Update	VPE	true	Normal	Normal
aP.VPEToolset.OE	• Open • Edit	VPE Toolset	true	Normal	Normal

Super User Group Permissions

Name	Action	Resource	Rule	Grant	Deny
aP. Component.Create.StrongGrant	Create	Component	true	Strong	Normal
aP. Component.RUD.StrongGrant	• Delete • Read • Update	Component	true	Strong	Normal
aP.Group.Create.StrongGrant	Create	Component	true	Strong	Normal
aP.Group.UD.StrongGrant	• Delete • Update	Group	true	Strong	Normal
aP.Permission.Associate. StrongGrant	Associate	Permission	true	Strong	Normal
aP.Group.Create.StrongGrant	Create	Permission	true	Strong	Normal
aP.Rollup.RUD.StrongGrant	• Delete • Read • Update	Rollup	true	Strong	Normal
aP.SystemAdmin.OE. StrongGrant	• Open • Edit	System Admin	true	Strong	Normal
aP.VPE.CostUsing.StrongGrant	Cost Using	VPE	true	Normal	Normal
aP.VPE.Create.StrongGrant	Create	VPE	true	Strong	Normal
aP.VPE.RUF.StrongGrant	• Delete • Read • Update	VPE	true	Strong	Normal
aP.VPEToolset.OE StrongGrant	• Open • Edit	VPE Toolset	true	Strong	Normal

Configure Admin Roles Using the Root Access Control Model

This example shows how to design and create these custom admin roles:

- User Admin – User admins provision aPriori Professional users.
- VPE Currency Admin – VPE currency admins modify aPriori Professional foreign exchange rates.

Prerequisites

Before you configure a custom role, define the requirements for the role explicitly and then convert the role requirements to permission requirements. Role requirements are basically a list of what a user in the role must be able to, or can, do and what the user must not be able to, or cannot, do in aPriori Professional.

Permission requirements express the role requirements in terms of actions, such as create, read, open, update, and edit. Permission requirements help you to define permission rules.

User Admins Requirements

The user admin role is for provisioning users in aPriori Professional. Therefore, user admins must be able to

- Create end users.
- Update end users.
- Remove end users.

User admins cannot open or edit any other features including:

- LDAP connections
- Roles
- License Modules
- Deployment – Default
- User Defined Attributes
- Dialog Views
- Permissions
- System Variables
- Composites Mapping Files

User admins can see admin users, but they cannot modify admin users.

Although user admins have limited abilities to modify groups, they cannot:

- Create groups.
- Change the name of admin groups or modify other properties of admin groups.
- Add users to system admin groups, VPE admins groups, or configured admin groups.
- Add or remove permissions that are associated to any group.
- Add, update, or delete group attributes.

Convert the role requirements into permission requirements. Group the permission requirements according to the actions that they grant or deny the right to do. For example, based on the User Admins role requirements, permissions for the group must:

- Deny the right to **Update** any admin groups including any configured admins group. (1)
- Grant the right to **Open** and **Edit**:
    - Group Membership (2)
    - Groups (3)
    - Users (4)

Therefore, four permissions (1, 2, 3, and 4) are needed for the User Admins group.

VPE Currency Admins Requirements

The VPE currency admin role is for modifying foreign exchange rates in aPriori Professional. Therefore, VPE currency user admins must be able to modify foreign exchange rates.

VPE currency admins cannot open or edit any of the other VPE features including:

- VPE Manage
- Cost Model Workbench
- Process Group Site Variable
- BOM Loader

Additionally, the VPE Currency Admins cannot read or modify these Deployment Data tab items:

- Process Groups
- Cost Taxonomy Display Names

Based on the VPE Currency Admins role requirements, permissions for the group must:

- Grant the right to **Open** and **Edit** foreign exchange rates. (1, 2)
- Deny the right to **Open** and **Edit** these VPE features:
    - VPE Manager
    - Cost Model Workbench
    - Process Group Site Variable
    - BOM Loader
- Deny the right to **Read** or **Modify**:
    - Process Groups
    - Cost Taxonomy Display Names

To enable the group to open and edit foreign exchange rates, you must grant permission to open and edit the **Currency** tab, which is a second-level tab in the VPE Toolset. The first-level parent tab to the **Currency** tab is the **Deployment Data** tab. Due to multigenerational permission interactions, to grant open and edit rights the **Currency** tab, you must also grant open and edit rights to the **Deployment Data** tab.

The aPriori Professional Root Access Control model does not include a permission that grants access to the **VPE Manager**, **Cost Model Workbench**, **Process Group Site Variable**, **BOM Loader**, **Process Group**, or **Cost Taxonomy Display Names** tabs. Therefore, you do not need to define a permission to deny access to these tools. You only need to define a permission if you want to grant access to them.

Therefore, only two permissions (1 and 2) are needed for the VPE Currency Admins group:

- Permission to **Open** and **Edit** rights to the **Deployment Data** tab
- Permission to **Open** and **Edit** rights to the **Currency** tab

Create Configured Groups and Define the Associated Permissions

After you define the requirements for the custom roles, create the groups, and define the permissions:

1. Create a group directory for the custom roles. The custom roles in this example are both configured administrator roles. Create one directory for the Configured Admins group, and then create sub-directories for the User Admins group and VPE Currency Admins group.

6.1. Open the System Admin Toolset. From the aPriori Professional menu bar, select **Tools > System Admin Toolset**.

6.2. Open the System Administrator. In the **System Admin Toolset** window, click the System Administrator,  , button.

6.3. Open the **Groups** tab. In the System Administrator window, click **Groups**.

6.4. Create a group. Under the Groups list, select the Add Group,  , button.

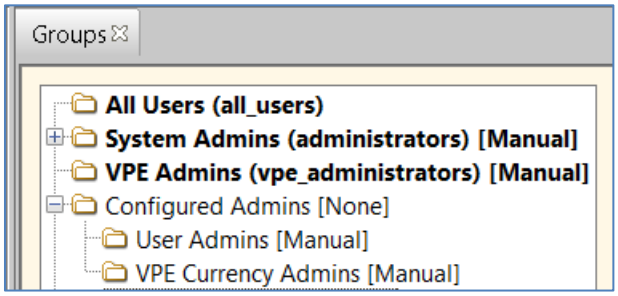
6.5. In the **Add Group** window, for:

6.5.1. **Group Name** – Enter *`Configured Admins`*.

6.5.2. **Group Membership** – Select `None`.

6.5.3. Click **OK.**

The new **Configured Admins** group directory appears in the **Groups** list.



6.6. Create sub-groups for the User Admin and VPE Currency Admin roles. Select the **Configured Admins** group directory and select the Add Sub-Group,  , button. for:

6.6.1. **Group Name** – Enter *`User Admins`*.

6.6.2. **Group Membership** – Select `Manual`.

6.6.3. Click **OK.**

6.6.4. Select the Add Sub-Group,  , button.

6.6.5. For **Group Name,** enter *`VPE Currency Admins`*.

6.6.6. For **Group Membership,** select `Manual`.

6.6.7. Click **OK.**

7. Define one of the four permissions that are required for the User Admins Group. The first permission denies the right to **Update** any admin groups including any configured admins group.

7.1. Open the **Permissions** tab. In the **System Administrator** window, click **Permissions.**

7.2. In the **Permissions** tab, create and configure the new permission:

7.2.1. Under the permissions list, select the Add,  , button.

7.2.2. In the Input tab, for **Permission Name** enter *`CA.Group.Update.NotAdmins`*. and then click **OK.**

7.3. In the Permissions tab, for:

7.3.1. **Description** – Enter *`Deny User Admins the right to add users to admin groups and the right to update admin groups.`*

7.3.2. **Resource** – From the dropdown menu, select **Group**.

7.3.3. **Action** – Select **Use, Update,** and **When this rule is true.**

7.3.4. **Rule** – Click **Edit** and enter:

```
(1 != (index(group.path, 'administrators/'))) &&
(group.name!='administrators') &&_

(1 != (index(group.path, 'vpe_administrators/'))) &&
(group.name!='vpe_administrators') &&_

(1 != (index(group.path, 'Configured Admins/'))) &&
(group.name!='Configured Admins')
```

7.3.5. Click **Edit** again**.**

7.3.6. **Grant –** Select **Normal: Actions are permitted unless blocked by a Strong Deny permission.**

7.3.7. **Deny –** Select **Normal: Actions are blocked unless permitted by another permission.**

For this rule, the table fields update automatically to:

(	Subject	Property	Operator	Subject	Property	And, Or,)
	CONSTANT	1	!=	FUNCTION	index(group.path, 'administrators/')	AND
	group	name	!=	CONSTANT	administrators	AND
	CONSTANT	1	!=	FUNCTION	index(group.path, 'vpe_administrators/')	AND
	group	name	!=	CONSTANT	vpe_administrators	AND
	CONSTANT	1	!=	FUNCTION	index(group.path, 'Configured Admins/')	AND
	group	name	!=	CONSTANT	Configured Admins	

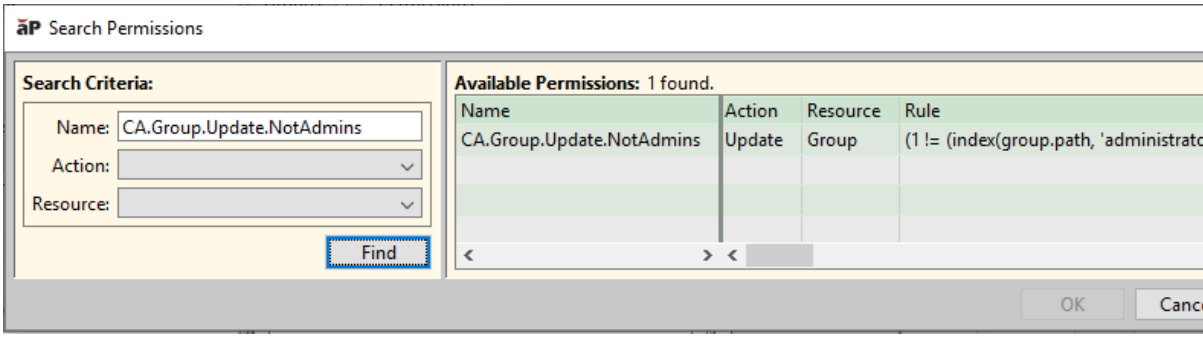
7.4. To save your changes, in the **System Administrator** toolbar, click the Publish, , button.

7.5. Add the newly defined permission to the User Admins Group.

7.5.1. In the **Groups** tab, in the groups list, select **User Admins [Manual]** group directory.

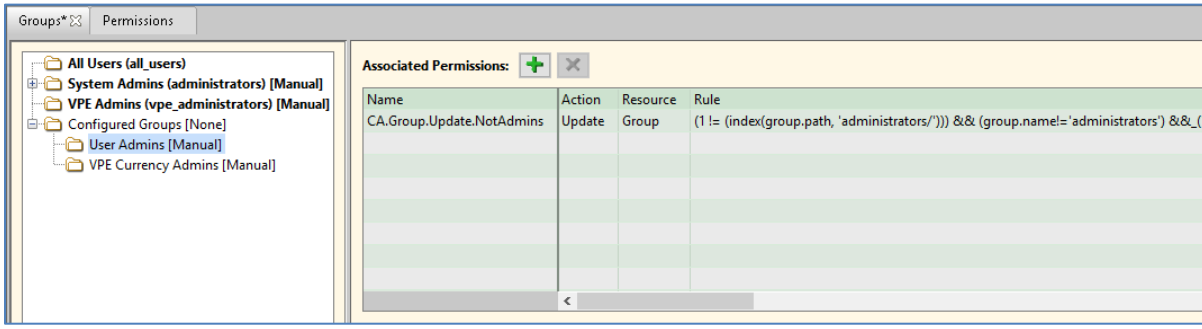
7.5.2. In **Associated Permissions,** select the Add Permissions to Group, , button.

7.5.3. In the **Search Permissions** window, for **Name,** enter `CA.Group.Update.NotAdmins` and then click **Find.**



The **CA.Group.Update.NotAdmins** permission appears in the list of **Available Permissions.**

7.5.4. In the **Available Permissions list,** select the **CA.Group.Update.NotAdmins** permission and then click **OK.**



The **CA.Group.Update.NotAdmins** permission appears in the list of **Associated Permissions** for the User Admins group.

8. Define the second permission that is required for the User Admins Group. The second permission grants the right to **Open** and **Edit** group membership for users.

8.1. In the **Permissions** tab, create and configure the new permission:

8.1.1. Click the Add, , button.

8.1.2. For **Permission Name, enter** `CA.SystemAdmin.GroupMembers.OE` and then click **OK.**

8.1.3. For **Description,** enter *Grant User Admins the right to open and edit user group memberships.*

8.1.4. For **Resource**, from the dropdown menu, select **System Admin**.

8.1.5. **Action** – Select **Use, Open, Edit,** and **When this rule is true.** For:

8.1.5.1. **Subject** – Select `uiElement`**.**

8.1.5.2. **Property** – `name` **is the only option.**

8.1.5.3. **Operator** – Select `==`**.**

8.1.5.4. **Subject** – Select **uiElementValue.**

8.1.5.5. **Property** – Select `groupMembers`

8.1.6. **Grant –** Select **Normal: Actions are permitted unless blocked by a Strong Deny permission.**

8.1.7. **Deny –** Select **Normal: Actions are blocked unless permitted by another permission.**

For this configuration, the generated rule is

```
uiElement.name==uiElementValue.groupMembers
```

8.1.8. To save your changes, in the System Administrator toolbar, click the Publish,  , button.

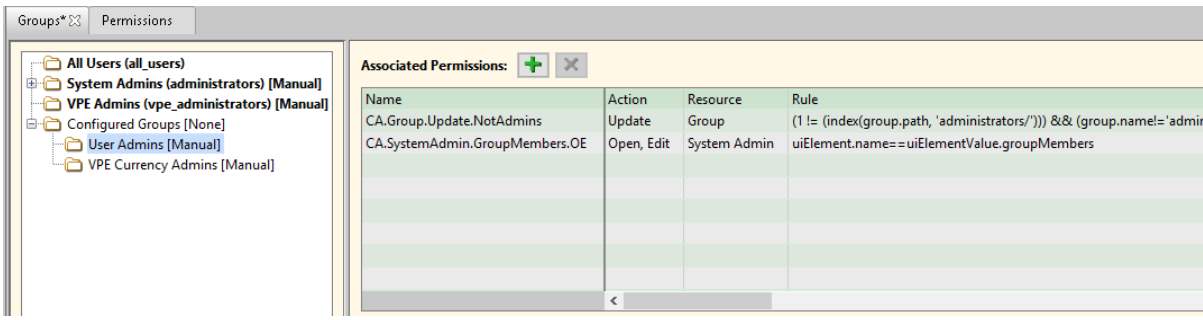
8.2. Add the newly defined permission to the User Admins Group.

8.2.1. In the Groups tab, in the groups list, select **User Admins** group directory**.**

8.2.2. In **Associated Permissions,** select the Add Permissions to Group,  , button.

8.2.3. In the **Search Permissions** window, for **Name**, enter *CA.SystemAdmin.GroupMembers.OE* and then click **Find.**

8.2.4. In the **Available Permissions** list, select the **CA.SystemAdmin.GroupMembers.OE** permission and then click **OK.**



The **CA.SystemAdmin.GroupMembers.OE** permission appears in the list of **Associated Permissions** for the User Admins group**.**

9. Define the third and fourth permissions that are required for the User Admins Group. These permission grants the right to **Open** and **Edit** groups and users.

9.1. In the **Permissions** tab, create and configure the new permission:

9.1.1. Select the **CA.SystemAdmin.GroupMembers.OE** permission that appears in the list of **Permissions and then** click the Copy, , button.

9.1.2. For **Permission Name, edit CA.SystemAdmin.GroupMembers.OE Copy** to *CA.SystemAdmin.Groups.OE* **and then click OK.**

9.1.3. For **Description, edit** *Grant User Admins the right to open and edit user group memberships* to *Grant User Admins the right to open and edit user groups.*

9.1.4. For **Action**, for the **uiElementValue Property,** select `groups`.

9.1.5. Select the **CA.SystemAdmin.GroupMembers.OE** permission that appears in the list of **Permissions** and then click the Copy, , button.

9.1.6. For **Permission Name,** edit **CA.SystemAdmin.GroupMembers.OE Copy** to *CA.SystemAdmin.Users.OE* and then click **OK.**

9.1.7. For **Description, edit** *Grant User Admins the right to open and edit user group memberships* to *Grant User Admins the right to open and edit users.*

9.1.8. For **Action**, for the **uiElementValue Property, select** `users`.

9.1.9. To save your changes, in the **System Administrator** toolbar, click the Publish, , button.

9.2. Add the newly defined permissions to the User Admins Group.

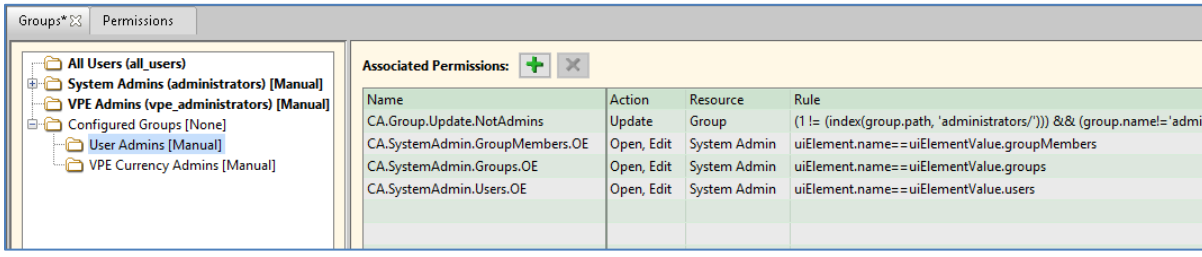
9.2.1. **In the Groups tab, in the groups list, select User Admins [Manual] group directory.**

9.2.2. In **Associated Permissions,** select the Add Permissions to Group, , button.

9.2.3. In the **Search Permissions** window, for **Name,** enter *CA.SystemAdmin* and then click **Find.**

9.2.4. In the **Available Permissions** list, select the **CA.SystemAdmin.Groups.OE** and the **CA.SystemAdmin.Users.OE** permissions and then click **OK.**

The CA.SystemAdmin.Groups.OE and **CA.SystemAdmin.Users.OE permissions** appear in the list of **Associated Permissions** for the User Admins group**.**



10. Define the permissions that are required for the VPE Currency Admins Group. The first permission grants the right to **Open** and **Edit** the Deployment Data tab in the VPE toolset. The second permission grants the right to **Open** and **Edit** the Currency tab in the VPE toolset.

10.1. In the **Permissions** tab, create and configure the new permission:

10.1.1. Click the Add,  , button.

10.1.2. **For Permission Name, enter** *CA.VPECurrencyAdmin.DeploymentData.OE* **and then click OK.**

10.1.3. **For Description, enter** *Grant VPE Currency Admins the right to open and edit the VPE Toolset Deployment Data tab.*

10.1.4. For **Resource,** from the dropdown menu, select **VPE Toolset**.

10.1.5. **Action – Select Use, Open, Edit, and When this rule is true. For:**

10.1.5.1. **Subject** – Select `uiElement`**.**

10.1.5.2. **Property** – `name` **is the only option.**

10.1.5.3. **Operator** – Select `==`**.**

10.1.5.4. **Subject** – Select **uiElementValue.**

10.1.5.5. **Property** – Select `deploymentData`

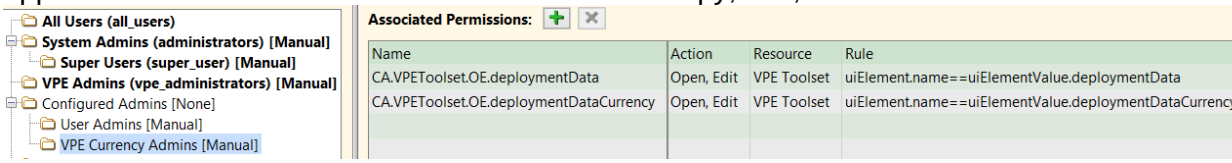
10.1.6. **Grant –** Select **Normal: Actions are permitted unless blocked by a Strong Deny permission.**

10.1.7. **Deny –** Select **Normal: Actions are blocked unless permitted by another permission.**

For this configuration, the generated rule is

```
uiElement.name==uiElementValue.deploymentData
```

10.1.8. Select the **CA.VPECurrencyAdmin.DataDeployment.OE** permission that appears in the list of **Permissions** and then click the Copy,  , button.



10.1.9. For **Permission Name,** edit **CA.VPECurrencyAdmin.DataDeployment.OE Copy** to *CA.VPECurrencyAdmin.DeploymentDataCurrency.OE* and then click **OK.**

10.1.10. For **Description,** edit *Grant VPE Currency Admins the right to open and edit the VPE Toolset Deployment Data tab* to *Grant VPE Currency Admins the right to open and edit the VPE Toolset Deployment Data tab Currency tab.*

10.1.11. For **Action**, for the **uiElementValue Property, s**elect `deploymentDataCurrency`.

10.1.12. To save your changes, **in the System Administrator toolbar,** click the Publish,  , button.

10.2. Add the newly defined permission to the User Admins Group.

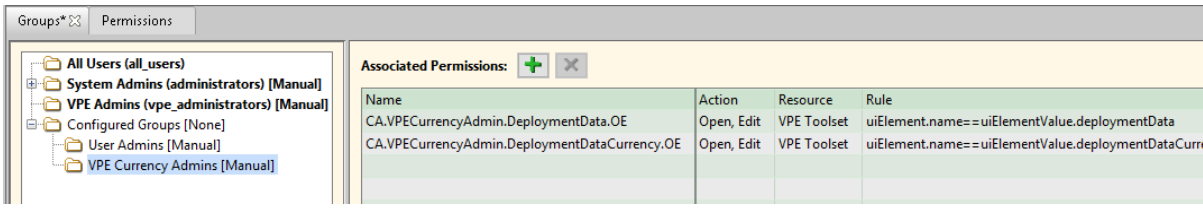
10.2.1. In the Groups tab, in the groups list, select **VPE Currency Admins** group directory.

10.2.2. In **Associated Permissions,** select the Add Permissions to Group,  , button.

10.2.3. In the **Search Permissions** window, for **Name,** enter *CA.VPE* and then click **Find.**

10.2.4. In the **Available Permissions** list, select the **CA.VPECurrencyAdmin.DataDeployment.OE** and the **CA.VPECurrencyAdmin.DataDeploymentCurrency.OE** permissions and then click **OK.**

The two new permissions appear in the list of **Associated Permissions** for the VPE Currency group.



10.3. To save your changes, in the **System Administrator** toolbar, click the Publish, , button.

6 Configuring the Geometry Analysis Tool (Heat Map)

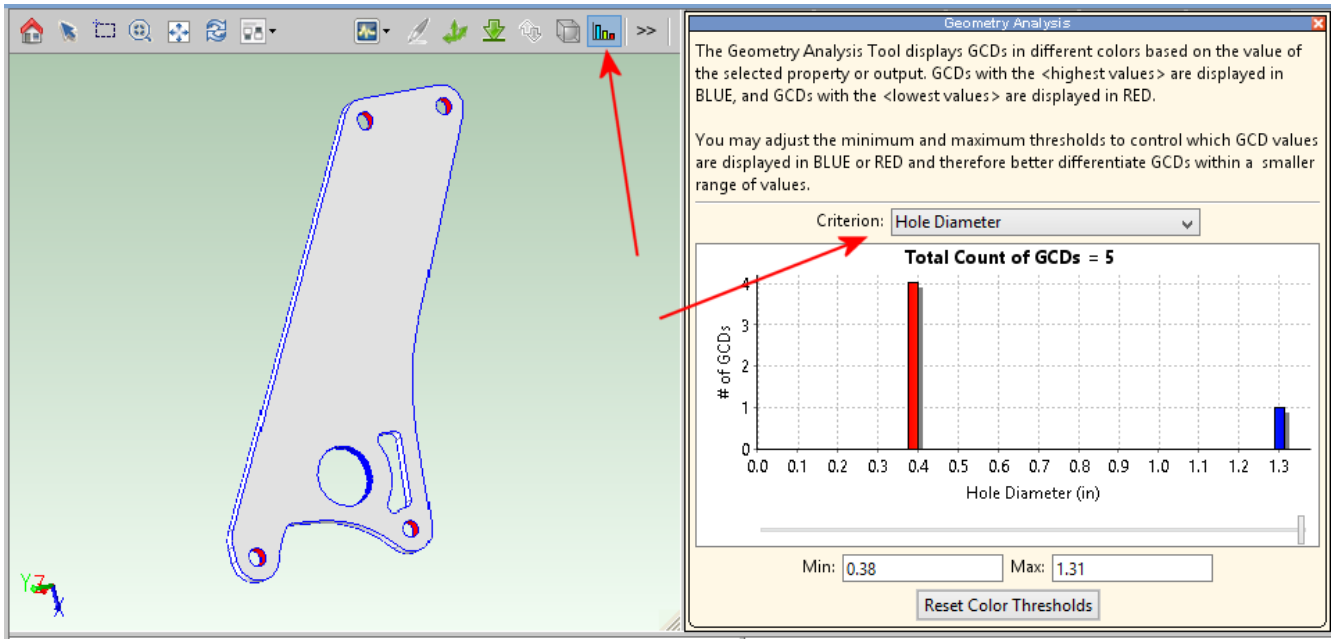
The aPriori “heat map” (more formally known as the “Geometry Analysis tool”) helps you visualize your component through the filter of various metrics. General usage of the heat map is covered in the aPriori *User Guide* and the *Cost Model Guide*. However, you can also configure the heat map and configure it to show information the way you want.

This chapter includes the following topics:

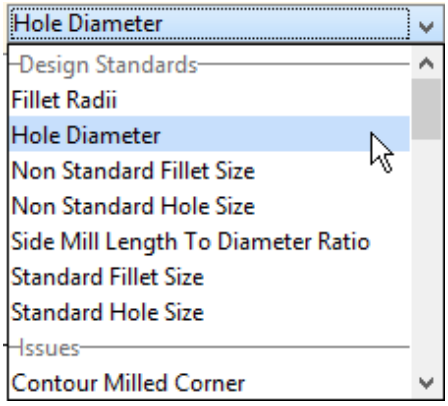
Heat Map Overview

The Geometry Analysis tool – better known as the “heat map” – is available from the component viewer tool bar and provides a visual presentation of metrics concerning various aspects of your machined component.

Note: In aPriori as shipped, the metrics are populated only for parts analyzed in the Stock Machining process group, but metrics for other process groups can be configured. Please contact aPriori support or your aPriori account team for more information.



aPriori comes out of the box with a rich set of heat map criteria which are available from the **Criterion** menu.



aPriori System Administrators have the ability to customize these criteria in the following ways:

- Edit criterion and category names.
- Reorganize criteria across categories or add them to new categories.

- Remove or suppress criteria.
- Set bounds and label for the X-axis,
- Add new criteria and make criteria available to other process groups other than Stock Machining (may require work in the Cost Model WorkBench; please contact aPriori support or your aPriori account team for more information).

Heat Map Configuration Basics

For basic configurations of the existing Heat Map menu criteria, you can edit the files found in:

```
<apriori_install>\ext\project-management-
   plugin\macro\app\ConfigureHeatMap
```

Note: As with any configuration procedure, MAKE SURE TO BACK UP ANY FILES BEFORE EDITING THEM.

In general, you:

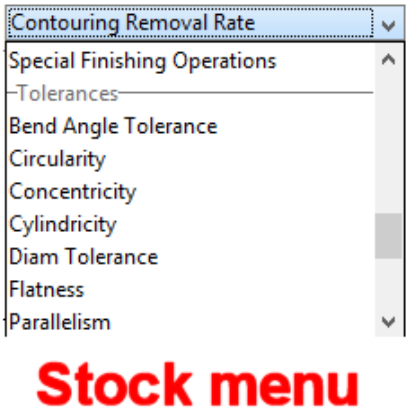
1. Make edits to `configHeatMap.xml`
2. Make supporting edits to one or more properties files in the `messages` sub-folder.

The examples provided in this section are not particularly realistic but give a simple introduction to heat map configuration.

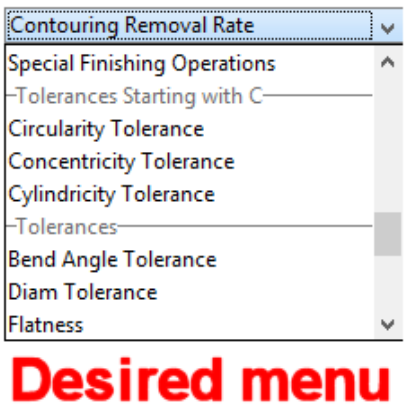
Heat Map Basic Edit Example

Assume that you have a manager who has been under a lot of stress lately and has started issuing some unusual orders. This week, he has stated that tolerances that begin with “C” (circularity, concentricity, and cylindricity) are exceptionally important and under-appreciated tolerances. To rectify this situation, he wants them to have their own grouping in the heat map **Criterion** drop-down menu. Further, he declares that they should have “Tolerance” appended to their names in the menu to give them the proper respect to which they are entitled.

Here is how the typical heat map drop-down menu appears, compared to how the manager wants to see it:



Stock menu



Desired menu

Here's how to make this change:

1 Go to *<apriori_install>*\ext\project-management-plugin\macro\app\ConfigureHeatMap and make a back-up copy of configHeatMap.xml.

2 Open configHeatMap.xml in an editor and scroll down to these lines:

```
<HeatMapToolSpec name="circularity">
   <GroupingSpecs>
         <GroupingSpec name="i18n(gtol)"/>
   </GroupingSpecs>
</HeatMapToolSpec>
<HeatMapToolSpec name="concentricity">
   <GroupingSpecs>
         <GroupingSpec name="i18n(gtol)"/>
   </GroupingSpecs>
</HeatMapToolSpec>
<HeatMapToolSpec name="cylindricity">
   <GroupingSpecs>
         <GroupingSpec name="i18n(gtol)"/>
   </GroupingSpecs>
</HeatMapToolSpec>
```

3 Modify these lines as shown in **bold** (don't miss the "_**c**" after "gtol"):

```
<HeatMapToolSpec name="circularity"
displayName="i18nMsg(circularity)">
      <GroupingSpecs>
   <GroupingSpec name="i18n(gtol_C)"/>
   </GroupingSpecs>
</HeatMapToolSpec>
<HeatMapToolSpec name="concentricity"
displayName="i18nMsg(concentricity)">
   <GroupingSpecs>
         <GroupingSpec name="i18n(gtol_C)"/>
   </GroupingSpecs>
</HeatMapToolSpec>
<HeatMapToolSpec name="cylindricity"
displayName="i18nMsg(cylindricity)">
   <GroupingSpecs>
         <GroupingSpec name="i18n(gtol_C)"/>
   </GroupingSpecs>
```

```
</HeatMapToolSpec>
```

Save this file when done.

4 Go to `<apriori_install>\ext\project-management-plugin\macro\app\ConfigureHeatMap\Messages` and make a back-up copy of `macroMessages.properties`.

5 Open `macroMessages.properties` in an editor and scroll down to this line:

```
HeatMap.gtol=Tolerances
```

6 Make copy of this line and edit it so that you end up with two lines as follows:

```
HeatMap.gtol=Tolerances
HeatMap.gtol_c=Tolerances Starting with C
```

7 Add the following lines at the end of the file:

```
HeatMap.circularity=Circularity Tolerance
HeatMap.concentricity=Concentricity Tolerance
HeatMap.cylindricity=Cylindricity Tolerance
```

Save the file when done.

8 Restart aPriori. The **Criterion** menu should now reflect the changes requested.

Configuring a “Favorites” section

Another way to make commonly used metrics easily accessible is to add a “favorites” grouping spec to the metric. For example, assume that the manager described in the previous example has an even stronger affinity for diameter tolerances than for tolerances starting with “c”. An aPriori administrator could make an additional edit to configHeatMap.xml to ensure that “Diam Tolerance” appears at the very top of the pull-down list, in the “Favorites” section:

1 Open `configHeatMap.xml` in an editor and scroll down to the “diamTolerance” section:

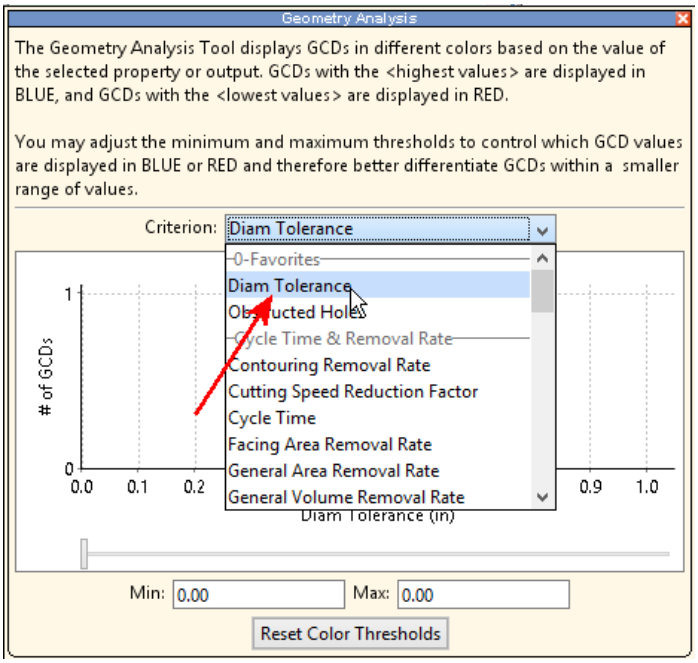
```
<HeatMapToolSpec name="diamTolerance">
      <GroupingSpecs>
            <GroupingSpec name="i18n(gtol)"/>
      </GroupingSpecs>
```

2 Add a “favorites” grouping spec line underneath the existing “gtol” line:

```
            <GroupingSpec name="i18n(gtol)"/>
            <GroupingSpec name="i18n(favorites)"/>
```

(You can optionally remove other “favorites” specs from other items if you want this particular item to stand put.)

3 Save the file when done, then restart aPriori. The favorites section of the **Criterion** menu should now prominently display the “Diam Tolerance” metric.



To suppress a heat map menu item

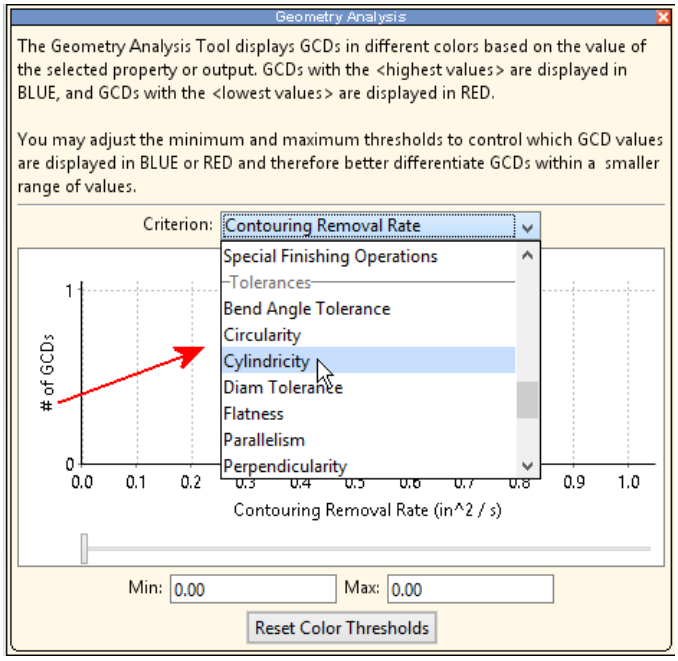
You can turn off the display of a heat map menu item using the **cslRuleForAction="false"** attribute setting. This can be combined with some condition statements to selectively turn off the menu item only under certain circumstances.

1 Open `configHeatMap.xml` in an editor and scroll down to the line defining the menu item you wish to suppress. For example, `<HeatMapToolSpec name="concentricity">`.

2 Insert the `cslRuleForAction` condition inside the closing angle bracket to suppress the line. For example:

```
<HeatMapToolSpec name="concentricity"
cslRuleForAction="false">
```

Save the file when done and restart aPriori. The “Concentricity” menu item is now missing from the list of tolerances.



3 To suppress the menu item for only a particular process group, add some conditions to the line. For example, to suppress this menu item except when there is a scenario open in the Sheet Metal process group::

```
<HeatMapToolSpec name=" concentricity " cslRuleForAction="{
false if (currentlyOpenScenario == null )  true if (
currentlyOpenScenario.processGroup.name == 'Sheet Metal'  ) )
false otherwise }" />
```

To control the X-axis display

You can set the minimum and maximum bounds for the heat map x-axis and set values to be displayed along the axis. For example, to customize the x-axis display for hole diameter:

1 Open `configHeatMap.xml` in an editor and scroll down to the line defining the menu item you wish to suppress. For example, `<HeatMapToolSpec name="holeDiameter" invertedGoodness="true" unit="Length" displayName="i18nMsg(holeDiameter)">`

Note: The "`invertedGoodness`" attribute is a Boolean that allows you to reverse the relative desirability display of an increasing value. In the world of cost management, increasing values are often negative: more cost, more waste, more material to be purchased, etc., and such increasing values are usually shown in the aPriori product with red arrows. However, sometimes increasing values represent a positive change: higher utilization, increased sales, etc. These can be represented by green arrows in the aPriori UI by setting `invertedGoodness="true"`.

2 Inside the closing angle bracket for this line, insert the bounds that you wish to use for the x-axis. For example, shown in **bold** below:

```
<HeatMapToolSpec name="holeDiameter" invertedGoodness="true"
unit="Length" displayName="i18nMsg(holeDiameter)"
lowerBound=".25" upperBound="2.0">
```

3 Inside the closing `</HeatMapToolSpec>` tag for this metric, add the labels you want to use along this axis. For example, when done the entire section should resemble:

```
<HeatMapToolSpec name="holeDiameter" invertedGoodness="true"
unit="Length" displayName="i18nMsg(holeDiameter)"
lowerBound=".25" upperBound="2.0">
   <GroupingSpecs>
         <GroupingSpec name="i18n(designStandards)"/>
   </GroupingSpecs>
   <XAxisLabelSpecs>
         <XAxisLabelSpec value=".3" label="i18n(smallHole)"/>
         <XAxisLabelSpec value=".75"
label="i18n(mediumHole)"/>
         <XAxisLabelSpec value="1.5" label="i18n(largeHole)"/>
   </XAxisLabelSpecs>
</HeatMapToolSpec>
```

Save the file when done.

4 Go to `<apriori_install>\ext\project-management-plugin\macro\app\ConfigureHeatMap\Messages` and make a back-up copy of `macroMessages.properties`.

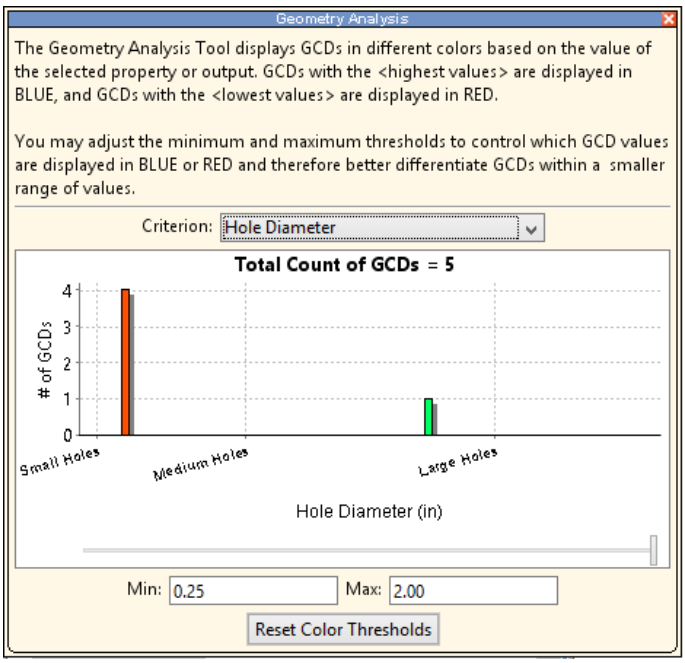
5 Open `macroMessages.properties` in an editor and scroll down to the end of the file.

6 Append the following lines to support the x-axis labels that you defined in `configHeatMap.xml`:

```
HeatMap.smallHole=Small Holes
HeatMap.mediumHole=Medium Holes
HeatMap.largeHole=Large Holes
```

Save the file when done.

7 Restart aPriori. The x-axis for Hole Diameter should now resemble the following:



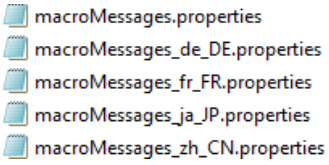
Heat map localization

The heat map is designed to support localization. The top section of `configHeatMap.xml` provides the following definition lines:

```
<messageBundles defaultBundle="macroMessages">
   <messageBundle baseName="macroMessages"
   filePath="messages/macroMessages.properties" country=""
   language="" variant=""/>
   <messageBundle baseName="macroMessages"
   filePath="messages/macroMessages_de_DE.properties" country="DE"
   language="de" variant=""/>
   <messageBundle baseName="macroMessages"
   filePath="messages/macroMessages_fr_FR.properties" country="FR"
   language="fr" variant=""/>
   <messageBundle baseName="macroMessages"
   filePath="messages/macroMessages_zh_CN.properties" country="CN"
   language="zh" variant=""/>
   <messageBundle baseName="macroMessages"
   filePath="messages/macroMessages_ja_JP.properties" country="JP"
   language="ja" variant=""/>
```

```
</messageBundles>
```

The `messages` subfolder contains a `macroMessages.properties` file, along with localized copies for each of the languages defined in `configHeatMap.xml`.



If your company has users in different locales, you should maintain configurations in parallel for all relevant languages.

7 Configuring the Wire Harness and PCBA Process Groups

The separately-licensed aPriori process groups for costing wire harnesses and PCBAs (printed circuit board assemblies) require that some configuration tasks be performed by an aPriori system administrator.

This chapter includes the following topics:

Wire Harness and PCBA Process Groups

The separately-licensed aPriori Wire Harness and PCBA Process Groups operate differently than most other cost models which are based on process groups that get their component information either from CAD models or from user-guided input.

The Wire Harness and PCBA process groups actually consist of special VPEs which include modified versions of the Assembly and the User Guided process groups. The components necessary to cost Wire Harnesses and PCBAs (such as VPE, UDAs, customized “macro” files, batch file scripts, etc.) are typically not shipped with the general aPriori release. Please contact aPriori Customer Support to acquire these components.

Before your users can use either of these process groups, somebody must perform the configuration steps documented in this chapter and in the "Wire Harness and PCBA VPEs" chapter of the *VPE Administration Guide*. This is typically done initially by aPriori Professional Services and then handed off to VPE and System Administrators at the customer site.

Note: Depending on the size of your site, the roles of System Administrator and VPE Administrator might be filled by the same person, or by different individuals. If you have any questions as to what steps need to be performed, or who should perform them, please contact your aPriori Professional Services representative.

Once these configuration steps have been completed, your users can use the feature as described in the "Wire Harness" and/or “PCBA” chapters of the aPriori *Cost Model Guide*.

Importing or Adding User Defined Attributes (UDAs)

The Wire Harness and PCBA Process Groups both require specific sets of User Defined Attributes (UDAs). You must add or import these UDAs before you or your users open the BOM Loader to cost the first wire harness or PCBA so that the field mappings can be set.

Note: UDAs can be exported and imported across aPriori installations but importing a set of UDAs **will wipe out any UDAs that currently exist in your system**. If you have existing UDAs that you wish to keep, you must *add* the following UDAs manually rather than by importing them.

Again, this procedure is typically done by aPriori Customer Service before delivering the process groups to you.

To add or import UDAs

1. In the aPriori client click **Tools> System Admin Toolset**.
2. Click the **System Administrator** icon on the System Admin Toolset window.
3. Select **User Defined Attributes** from the navigation pane on the left.

If UDAs **do** appear in the main panel, you *must* manually enter each UDA as described in the next step below, so that you do not delete your existing UDAs.

If no UDAs appear in the main panel, you can safely import the required UDA set as described in the two steps below.

1 *To ADD UDAS manually:* Manually enter each UDA as shown in the images below (one for Wire Harness, one for PCBA, as necessary). For each UDA, click the first empty row in the Name column and type the name exactly as shown. Then select the Type column to specify **String** or **Number** from the drop-down menu. When done, click the **Publish Changes** icon () in the toolbar when done.

Wire Harness UDAs:

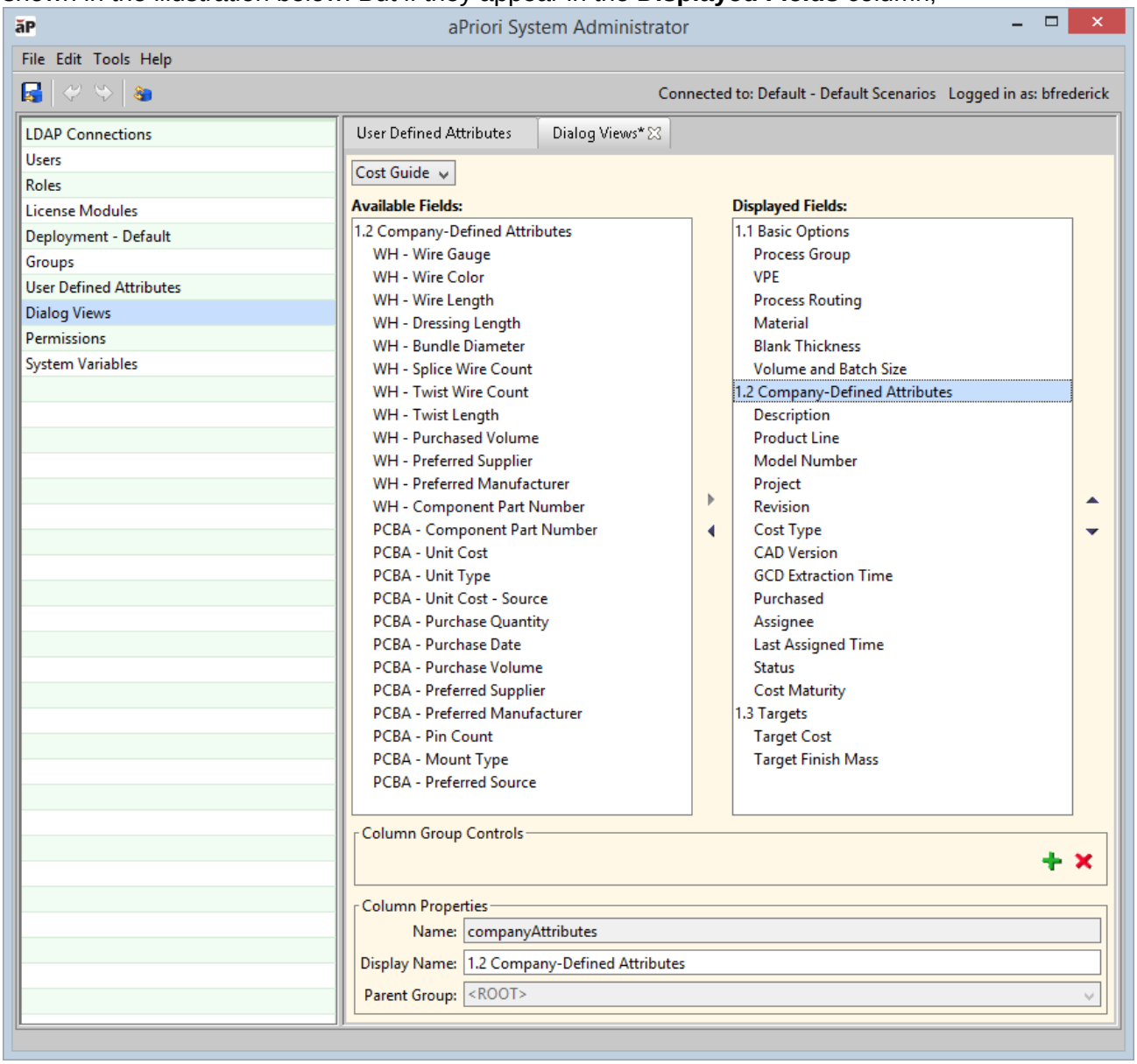
Name	Display Name	Type
PCBA_COMPONENT_PN	PCBA - Component Part Number	String
PCBA_MANUALBOM_UNIT_COST	PCBA - Unit Cost	Number
PCBA_MANUALBOM_UNIT_TYPE	PCBA - Unit Type	String
PCBA__MANUALBOM_SOURCE	PCBA - Unit Cost - Source	String
PCBA_MANUALBOM_UNIT_QTY	PCBA - Purchase Quantity	Number
PCBA_MANUALBOM_PURCHASE_DATE	PCBA - Purchase Date	Date
PCBA_PURCHASED_VOLUME	PCBA - Purchase Volume	Number
PCBA_PREFERRED_SUPPLIER	PCBA - Preferred Supplier	String
PCBA_PREFERRED_MANUFACTURER	PCBA - Preferred Manufacturer	String
PCBA_PIN_COUNT	PCBA - Pin Count	Number
PCBA_MOUNT_TYPE	PCBA - Mount Type	String
PCBA_PREFERRED_SOURCE	PCBA - Preferred Source	String

PCBA UDAs:

Name	Display Name	Type
PCBA_COMPONENT_PN	PCBA - Component Part Number	String
PCBA_MANUALBOM_UNIT_COST	PCBA - Unit Cost	Number
PCBA_MANUALBOM_UNIT_TYPE	PCBA - Unit Type	String
PCBA__MANUALBOM_SOURCE	PCBA - Unit Cost - Source	String
PCBA_MANUALBOM_UNIT_QTY	PCBA - Purchase Quantity	Number
PCBA_MANUALBOM_PURCHASE_DATE	PCBA - Purchase Date	Date
PCBA_PURCHASED_VOLUME	PCBA - Purchase Volume	Number
PCBA_PREFERRED_SUPPLIER	PCBA - Preferred Supplier	String
PCBA_PREFERRED_MANUFACTURER	PCBA - Preferred Manufacturer	String
PCBA_PIN_COUNT	PCBA - Pin Count	Number
PCBA_MOUNT_TYPE	PCBA - Mount Type	String
PCBA_PREFERRED_SOURCE	PCBA - Preferred Source	String

2 *To IMPORT UDAs:* If aPriori has provided you with a UDA import file, and if you do not have any existing UDAs, click **File > Import > User Defined Attribute Data** and navigate to the UDA import file. (Remember, importing UDAs will remove any existing UDAs you might have.) After importing, your main panel should look like the illustration above. When done, click the **Publish Changes** icon () in the toolbar when done.

3 Before exiting, you should ensure that these new UDAs are not set to display in the Cost Guide. Click **Dialog Views** in the navigation pane and select **Cost Guide** from the pull-down menu. It is okay if the UDAs appear in the **Available Fields** column as shown in the illustration below. But if they appear in the **Displayed Fields** column,



Adding Costing Macros to Your Installation

The User Interface (UI) for Wire Harness and PCBA is implemented with aPriori *macros* – custom modules that extend the capability of the out-of-box aPriori Professional product. Installation and configuration of these macros must be done with the assistance of aPriori Professional Services. However, the basic steps include:

- Copying files provided by aPriori Professional Services into the `project-management-plugin` folder underneath the aPriori Professional installation folders.

- Editing those files as directed by aPriori Professional Services.
- Publishing those changes to make them available to all users.

8 Using CAD Properties

Most CAD systems provide the ability to define and set the value of properties (sometimes called parameters) that are stored with the CAD file. These properties can capture valuable information about the component. Many customers are using these properties as part of Model Based Definition (MBD) initiatives, to store additional information relevant to the manufacturing, inspection, and analysis of the component.

aPriori provides the ability to extract these CAD properties and use them to automate cost estimation. For supported CAD systems, all inputs necessary to cost a part can be supplied automatically from the CAD file, reducing errors and enabling faster and more accurate costing. CAD Properties can also be mapped to aPriori User Defined Attributes (UDAs).

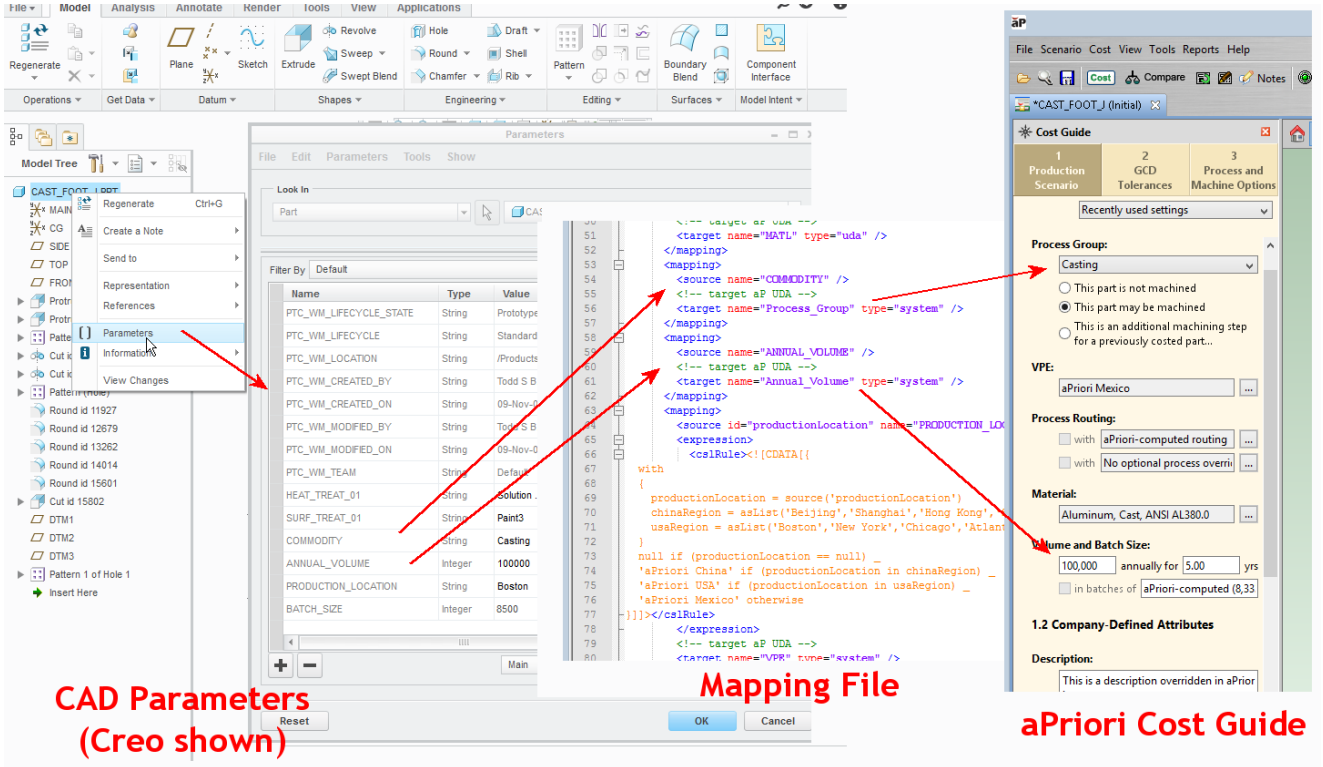
This chapter includes the following topics:

Overview of CAD properties

CAD files often contain valuable information stored as *properties* or *parameters* (the term varies from product to product). Some examples of these properties include:

- material
- description
- surface and heat treatment requirements

The CAD properties or parameters can be mapped to aPriori User Defined Attributes (UDAs) and to certain system-defined production input attributes using an XML mapping file. The following illustration shows an example of mapping two production inputs (Process Group and Annual Volume).



In the past, the mappable production input attributes have included Material, Description, and Revision. As of Release 2018 R1 SP1, this has been extended to also include Process Group, VPE, Annual Volume, and Batch Size, meaning that now ALL inputs required for costing a part can be provided by CAD properties.

A material property can automatically drive aPriori Material. A description property can contain meaningful text that complements the component part number. A surface or heat treatment requirement property could be mapped to a UDA to indicate that aPriori needs to add a secondary process.

aPriori extracts properties from CAD files whenever it extracts (or re-extracts) GCDs. All properties are extracted and stored in the aPriori database with the GCDs, but with one exception only *mapped* properties are visible to aPriori users. (The exception is CAD standard CAD material properties, which will automatically get mapped to the aPriori material property IF it exactly matches an aPriori material.)

Properties get mapped to UDAs and production inputs by means of a user-written XML file. This XML mapping file is prepared externally and then is loaded and stored in the aPriori database from the System Administration menu item **File > Import > CAD Property Mapping File**.

Notes:

- Because every company has different CAD property fields, aPriori cannot provide a standard Out-of-Box mapping file. However, we have provided some examples in this chapter to illustrate the concepts discussed here. See *CAD Properties Mapping Examples*. Please contact aPriori Support or Professional Services to acquire an example starting point mapping file and guidance for how to modify it for your purposes.
- In releases prior to 2018 R1 SP1, the mapping file was loaded with: **File > Load CAD Property Mapping File**. The new menu location now is: **File > Import > CAD Property Mapping File**. This is paired with a new menu item to save a copy of the currently loaded mappings: **File > Export > CAD Property Mapping File.**

Supported CAD Systems and Neutral File Formats

aPriori's property extraction feature supports the following CAD systems:

- Pro/ENGINEER and CREO (includes Direct Integration mode)
- CATIA (includes the “Cost-in-aPriori” Plug-In)
- NX (includes the “Cost-in-aPriori” Plug-In)
- SolidWorks (includes the “Cost-in-aPriori” Plug-In)
- STEP AP242

Notes:

1. Earlier versions of the STEP standard (AP203 and AP214) do not support CAD properties.

2. aPriori supports property extraction from part- and assembly-level but does not currently support feature-level attributes.

3. aPriori supports property extraction from a Family-Table Instance of a component but not a generic.

4. aPriori does not support extracting SOLIDWORKS properties that are defined in configurations.

5. NX properties are sometimes defined as “category/attribute”. This can be mapped in aPriori using the vertical bar (“|”): <source name="**MyCategory|MyAttribute**"/>

Summary of Enhancements in 2018 R1 SP1

If you have been using the original release of CAD Properties (introduced in aPriori Professional Release 2015 R1), the following list of enhancements and changes will help you to take advantage of the new functionality introduced in Release 2018 R1:

- The supported aPriori system-defined attributes now include the following production inputs: Process Group, VPE, Annual Volume, and Batch Size. (This means that all inputs necessary to cost a part can now be supplied automatically from the CAD file.)
- The CAD Property Mapping File format introduced in aPriori 2015 R1 has been extended to support the new features in 2018 R1 SP1.
- You can now include CSL expressions in your mapping file, meaning that it is no longer necessary for the value of the CAD property to exactly match an available value of an aPriori field. For example, the CAD field COMMODITY with a value of “Casting” could be mapped to the “Casting – Die” process group in aPriori. This also means that the "Material" restriction in earlier versions is no longer necessary: CAD properties specifying materials no longer need to use terminology 100% the same as aPriori.
- The availability of CSL expressions in the mapping file also means that the value of the aPriori field can be driven based on the value of multiple CAD fields as well as just a single CAD field. For example, the aPriori VPE may be chosen based on the values of both the PRODUCTION_REGION and TOOLING_REGION CAD fields.
- A new export feature allows you to create a fresh XML file from the currently loaded mappings, including comments. Previously there was no way to know for sure what mapping file was in effect, and no way to confirm that an XML mapping on disk was exactly the same as the one that was last loaded. This new feature is found under the System Administration **File > Export** > **CAD Property Mapping File** menu item. Also note that the original import mechanism (**File > Load CAD Property Mapping File**) is now found at **File > Import > CAD Property Mapping File**.
- Values mapped from CAD properties can now be viewed and cleared in aPriori, to explore the cost impact of different manufacturing scenarios. See the aPriori Professional *User Guide* for more information about the **Cost Overrides** dialog or the **Reconcile Inputs** dialog.

Production Input Mappings

The following aPriori out-of-box (system-defined) production inputs can be mapped to CAD properties. Note that in the XML file, you must use the system-defined name, NOT the display name, in those cases where they differ.

System Name (MUST be used in XML mapping)	Display Name
Process_Group	Process Group
VPE	VPE
Material	Material
Annual_Volume	Annual Volume
Batch_Size	Batch Size
Description	Description

Revision	Revision

In the mapping file, these standard, out-of-box production inputs are defined as type "system". No other aPriori system-defined attributes can be mapped to CAD properties.

For information about mapping CAD properties to aPriori User Defined Attributes (UDAs), see the next section, *UDA Mappings*. Note that it is possible to create UDAs with the same names as system-defined production attributes. If you do, you can map CAD properties to either the UDA, or to the production input, or to both. And you can display either or both on the Cost Guide.

Of the mappable production input attributes, Material and Description exhibit some special-case behavior as described below.

Description

When mapped to a CAD property, the Description system-defined attribute in aPriori is set to the value of its corresponding CAD property only when the CAD model is first opened and GCDs are first extracted. It will not appear in the Cost Overrides dialog and will not be subsequently updated from the CAD property again. This ensures that if this is modified in aPriori, the modified value will not be overwritten by the CAD property.

Note that this behavior may change in a future release.

Material

Each CAD system typically provides a mechanism for specifying material, and the name of the chosen material is typically stored or exposed as a standard property or parameter. For example, Pro/ENGINEER stores this information in "PTC_MATERIAL_NAME". aPriori can read these material specifications and attempts to map them to aPriori materials. Prior to Release 2018 R1 SP1, the CAD material property value had to match an aPriori material exactly. As of 2018 R1 SP1, the XML mapping file can now use CSL to map the CAD property to an aPriori material (i.e., the CAD material property no longer needs to exactly match an aPriori material name, but the output of the CSL mapping expression does). If an exact match cannot be made, the VPE default material is used.

Note that a similar capability has been available for some time in Direct Integration mode for Pro/ENGINEER. CAD property extraction now makes this available for all supported CAD systems in CAD Independent mode and plug-ins.

aPriori material mapping also provides the ability to use a “generic” CAD property to influence material selection in aPriori. Some sites use a different property other than the standard one to store the material name (for example, a Pro/ENGINEER site might use “MATERIAL_DETAILS” instead of “PTC_MATERIAL_NAME”).

A Note About PLM Mappings

Properties such as "Revision" are often driven by a Product Lifecycle Management (PLM) system that your company uses in concert with your CAD software. For example, in a PTC installation, the Creo CAD system may be coordinated with a Windchill PLM system.

aPriori does not currently extract properties directly from PLM systems. However, you may be able to configure your PLM/CAD installation so that certain PLM properties are

“pushed” into the CAD file each time they are updated. If your installation is configured this way, and the PLM properties are stored in the CAD file each time they are updated, then aPriori can access them from the CAD file.

UDA Mappings

CAD properties can also be mapped to aPriori User Defined Attributes (UDAs). In the mapping file, these targets are defined as type "uda". As mentioned earlier, a CAD property can be mapped to a UDA, a production input ("system"), or to both.

Mapping UDAs that have default values

Caution: The mapping behavior for UDAs is basically the same as that for system-defined production inputs. However, if you have defined a default value for an aPriori UDA through the System Administration UDA dialog, that UDA will NEVER use a CAD Property value, even if it is correctly mapped. (This behavior may change in a future release.)

For more information about UDAs, see *Managing User Defined Attributes (UDAs)*.

General Rules for Mapping CAD Properties

The following information applies to both UDAs and production inputs.

Multiple Mappings and Precedence

It is possible to map more than one CAD property to a UDA or a production input. This handles the situation where more than one CAD property for the same attribute exists and is populated in the CAD file. For example, a company's use of CAD properties might evolve over time, so that different properties may be present in parts of different ages or from different organizations within the company. Or, the CAD file may specify both a "Material" and an "Alternate Material" field. Precedence in aPriori is determined by the order in which the mappings occur in the XML file: the first mapping that succeeds "wins".

Overriding Mapped Values

Your users can override mapped values in aPriori to explore the cost impact of different manufacturing scenarios.

You can override mapped values in the following ways:

- Enter a different value in the **Cost Guide** (or in the **Cost Object Info** dialog).
- Specify a different value in the Bulk Costing user interface or spreadsheet

Once a mapped value is overridden, the override value continues to be used for subsequent costings, unless it is cleared. There are two ways to clear an override:

- Use the **Cost Overrides Summary** dialog

- Using the **Reconcile Inputs** dialog.

Elements of an XML CAD Mapping File

To map CAD properties to aPriori UDAs and production inputs, you must:

- define these mappings in an XML file
- verify that the XML is valid
- load the file into aPriori

Note: Because every company has different CAD property fields and different requirements, aPriori does not provide a standard out-of-box mapping file. We suggest that you contact aPriori Customer Support or Professional Services for a starting file that is appropriate for your company. Every XML mapping file must consist of a series of source/target mapping statements, where the source specifies the CAD property and the target specifies the aPriori attribute and its type. We advise against trying to copy and paste snippets found in the documentation into a mapping file, since formatting and autocorrected characters like straight quotes can cause problems that are difficult to debug.

Note: CAD properties and aPriori attributes are case-sensitive.

Here is a sample of a mapping file from Release 2015 R1. It defines different mappings for PRO/E and NX systems, for both User Defined Attributes and production inputs. The Pro/E Material mapping maps multiple properties to a single Material attribute. In this case, the first mapping that works takes precedence.

The steps that you would follow to create this file are explained later in this chapter, in *CAD Properties Mapping Examples*. Additional examples showing some of the capabilities introduced in Release 2018 R1 SP1 are also provided.

```
<?xml version="1.0" encoding="utf-8"?>
<database>
  <mappings>
    <cadPropertyMapping>
      <!-- may specify multiple modelers and model type -->
      <modelFilter modelerTypes="PROE" modelType="PART"/>
      <mapping>
        <!-- source CAD property-->
        <source name="MODELED_BY"/>
        <!-- target aP UDA -->
        <target name="Designer"/>
      </mapping>
      <mapping>
        <!-- source CAD property-->
        <source name="DESCRIPTION"/>
        <!-- target aP system property -->
        <target name="Description" type="system"/>
      </mapping>
      <mapping>
        <source>
           <name>MATERIAL_OVERRIDE</name>
           <name>PTC_MATERIAL_NAME</name>
        </source>
        <target name="Material" type="system"/>
      </mapping>
    </cadPropertyMapping>
    <cadPropertyMapping>
      <modelFilter modelerTypes="NX"/>
      <mapping>
        <!-- source CAD property-->
        <source name="DEVELOPER"/>
        <!-- target aP UDA -->
        <target name="Designer"/>
      </mapping>
      <mapping>
        <!-- source CAD property-->
        <source name="REFERENCE_COMPONENT"/>
        <!-- target aP system property -->
        <target name="Description" type="system"/>
      </mapping>
      <mapping>
        <!-- source CAD property-->
        <source name="MaterialPreferred"/>
        <!-- target aP system property -->
        <target name="Material" type="system"/>
      </mapping>
    </cadPropertyMapping>
  </mappings>
</database>
```

CAD Mapping Tag Reference

The following table lists the available XML mapping tags as of Release 2019 R1 SP1. Closing tags are not listed.:

TAG	ATTRIBUTE(S)	DESCRIPTION
<database>		Top-level required tag.
<mappings>		Second-level required tag
<cadPropertyMapping>		If you have multiple CAD systems and the mappings are the same for all of them, you can have a single <cadPropertyMapping> section with a <modelFilter> tag (see below) that specifies multiple "modelerTypes" attributes. For different mappings, use multiple <cadPropertyMapping> sections, each with its own <modelFilter> tag specifying the "modelerType" attribute for the specific CAD system.

<modelFilter>	modelType
	
	modelerTypes
		Defines the kind of model and what CAD system produced it, as well as the path to the file (optional).
		
		Valid values are:
		ASSEMBLY
		PART
		
		Valid values can be any combination of space-separated:
		PROE
		CATIA
		NX
		SOLIDWORKS
		and/or
		STEP
		For example:
		`<!-- may specify multiple modelers, model type and cadFileRegEx -->`
		`<modelFilter modelerTypes="PROE CATIA NX STEP" modelType=" PART ASSEMBLY">`

<mapping>		Contains a source/target mapping pair.
<source>	name	Each <source> tag much specify at least one named CAD property to be mapped. You can

		specify the name by attribute, or you can use the <name> tag (see below), particularly if you wish to map to multiple properties. If you specify multiple <name>tags, the first one to match takes precedence. If you specify both a name attribute and one or more <name> tags within a <source> specification, the attribute takes precedence. **Note:** For NX attributes that are defined under a category, use the vertical bar ("|") to separate the category and attribute. For example: <source name="**MyCategory|MyAttribute**"/>
	id	The id attribute was added in 2018 R1 and is only needed if your <mapping> element needs to access the <source> via a CSL expression. See also the <expression> tag.
<target>	name	Name of the aPriori attribute to be mapped. Note: This is NOT the display name.
	type	The type of aPriori attribute to be mapped. Valid values are: uda system If not specified, "uda" is assumed.
<name>		Same as the <source> "name" attribute (see above). If you use multiple <name> tags within a <source> tag (for mapping multiple properties to a single aPriori attribute), the first match takes precedence. If you specify both a name attribute and one or more <name> tags within a <source> specification, the attribute takes precedence. Therefore, the following statements are equivalent: <source name="Tooling Region"> <name>TOOLING_REGION</name> </source> <source> <name>Tooling Region</name> <name>TOOLING_REGION</name> </source>

<expression> **<cslRule>** **<![CDATA[** *CSL code* **]]**		These tags are used to support CSL expressions in the mapping file. This capability was introduced in aPriori Professional 2018 R1. See also the id attribute for the <source> tag. Note that "cslRule" can also be specified as an attribute to the expression tag (for example, "<expression cslRule="*csl code*…"). However, for clarity aPriori recommends that you use individual <cslRule> tags rather than attributes. For more information and examples, see the sections starting at *Using CSL Logic in Your Mappings*.

Importing a CAD Property Mapping File

Once you have created a mapping file, you load it from the System Administration window by clicking **File > Import > CAD Property Mapping File.** This command prompts you to browse the operating system directory structure and navigate to the mapping file you want to load into the aPriori database.

Exporting a CAD Property Mapping File

To save a copy of the CAD property file that is currently loaded, use **File > Export > CAD Property Mapping File.** This command prompts you to specify a folder and an XML file name in which to save the current mappings.

Clearing Your Mappings

You can clear out your mappings by importing a correctly-formatted, near-empty mapping file with the following contents:

```
<?xml version="1.0" encoding="utf-8"?>
<database />
```

CAD Properties Mapping Examples

This section describes the steps of configuring and using a CAD properties mapping file:

- Examining your CAD file to identify properties to be mapped
- Identifying aPriori production inputs, and/or adding UDAs to aPriori to be mapped to these properties and making them visible on the Cost Guide.
- Creating an XML file to map properties to UDAs and production inputs.
- Testing the extraction process

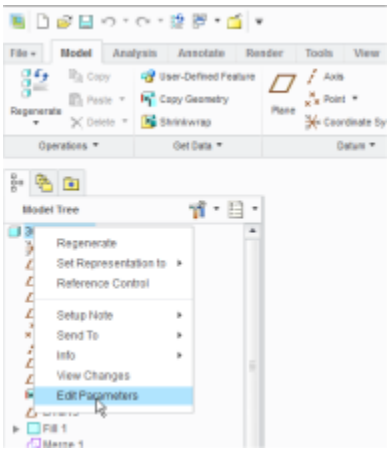
Examining Your CAD File to Identify Properties To Be Mapped

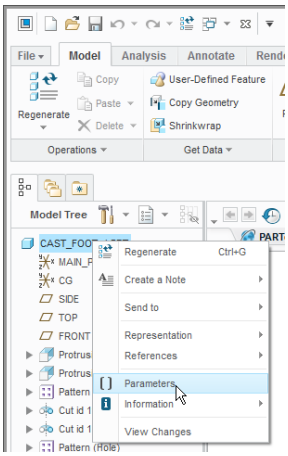
Before you map your CAD properties (also referred to as parameters) to aPriori, you need to know which ones you want to map. CAD systems that support properties provide a way to view and modify them. The specifics will vary from system to system but should be fairly easy to identify. This example describes PTC Creo parameters. Adjust the example for your particular CAD installation.

To identify CAD properties to be mapped

1 Start your CAD system and open a part that contains properties that you wish to map for aPriori.

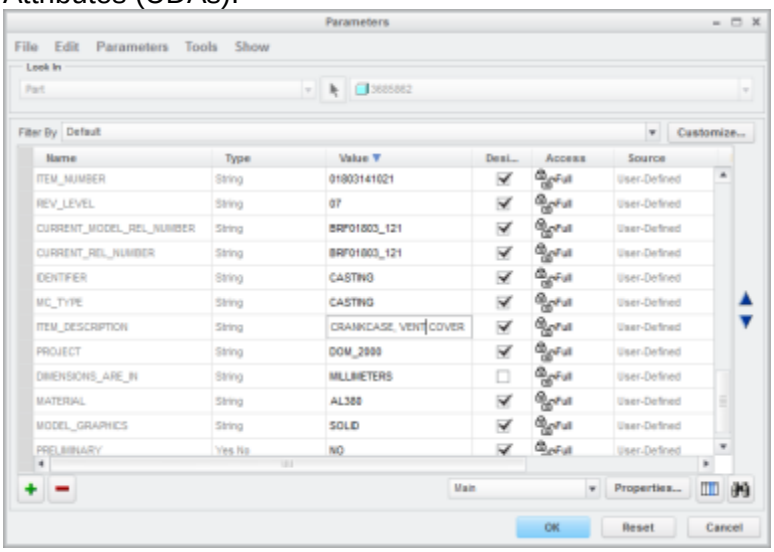
2 Display the part properties. This example assumes that the CAD system is PTC Creo, in which case you would right-click the top-level entry of the Model Tree and click **Edit Parameters** or **Parameters** (depending on Creo version) on the context menu. Adjust this step for your CAD system





3 In the resulting properties display, review the available properties and make a list of the ones that you want to map for aPriori. Also note which, if any, should be mapped to aPriori system-defined production inputs (Process Group, VPE, Material, Description, Revision, Annual Volume, Batch Size) rather than to User Defined

Attributes (UDAs).



In this example, we will map the Creo "ITEM_DESCRIPTION" parameter to the aPriori "Description" system-defined attribute, and the Creo "MATERIAL" parameter to the aPriori "Material" system-defined attribute. We will also plan to map the following Creo parameters to aPriori User Defined Attributes that we will define in the next section: REV_LEVEL, CURRENT_REL_NUMBER, and PROJECT.

Note that "PROJECT" is a bit of a special case: there is an aPriori system-defined attribute named "PROJECT", but it is not one of the production inputs that can be mapped to a property. So, we will map this to a new UDA named "PROJECT_ATTR" just to demonstrate that you can have UDAs and project inputs with the same name, and to caution you about this situation when planning your mappings.

Also note that this example maps the CAD "REV_LEVEL" property to a UDA named "REV_LEVEL". You could also have mapped this (or a different CAD property named "REVISION" if it existed) to the aPriori system-defined attribute "REVISION". The XML for this alternative mapping is shown commented out in the XML screenshot in the previous section.

Adding UDAs to aPriori

Now that you know which CAD properties you want to map; you need to ensure that there are corresponding attributes in aPriori for them.

To demonstrate various aspects of property mapping, we will give one UDA the same name as its corresponding CAD property, and different names for the others. We will use mixed-case and spaces for displaying them in the UI.

For more general information about UDAs, see: *Managing User Defined Attributes (UDAs)*.

This table shows the properties and attributes used in this example:

CAD Property	aPriori Attribute	aPriori Display Name	aPriori Type	Comment
ITEM_DESCRIPTION	Description	Description	System	Does not need to be created, but

				does need to be mapped
MATERIAL	Material	Material	System	Does not need to be created but does need to be mapped. Also requires additional configuration.
PROJECT	PROJECT	Project_2	UDA	There is a non-mappable aPriori system-defined attribute named PROJECT, so need to create a new UDA.
CURRENT_REL_NUMBER	Current_Rel	Current Release	UDA	Create a UDA with a different name and map it in the XML file.
REV_LEVEL	REV_LEVEL	Rev Level	UDA	Create a UDA with the exact same name as the property and map it in the XML file..

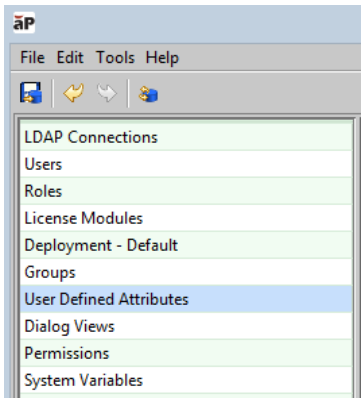
To add UDAs

Follow the same procedure as you would to create any UDA (see *Managing User Defined Attributes*).

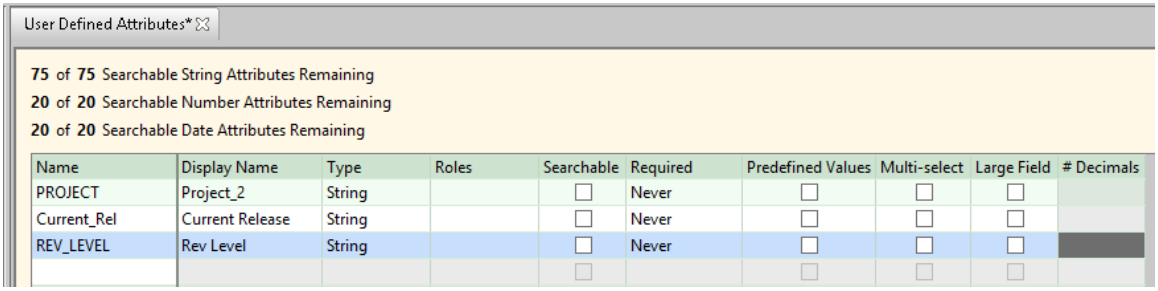
1 In the aPriori client, click **Tools > System Admin Toolset**.

2 Click the **System Administrator** button ( ) on the resulting window.

3 Click **User Defined Attributes** in the System Administrator window.

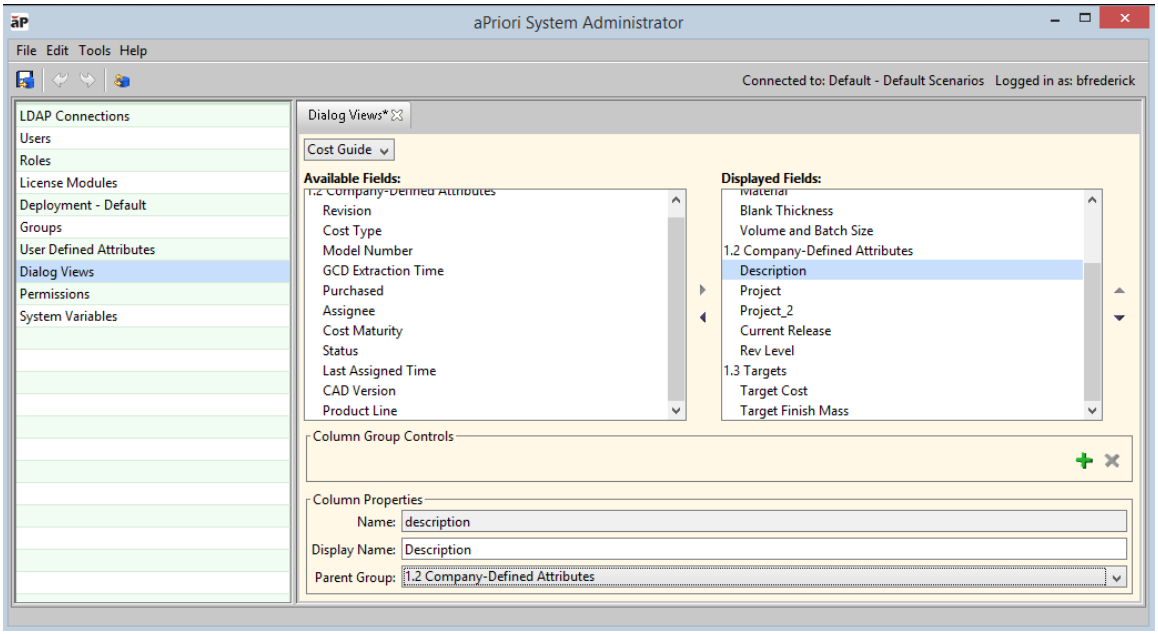


4 In the User Defined Attribute pane, double-click the fields to enter information from the table above. (Remember, ITEM_DESCRIPTION and MATERIAL are aPriori system-defined production inputs that do not need to be created as UDAs.) When done, the results should look like the screenshot below.



5 When done, click the **Publish Changes** button ( ) in the toolbar.

6 Now that the UDAs exist, you can adjust their appearance (if necessary) on the Cost Guide. By default, the new UDAs will appear in section "1.2 Company-Defined Attributes". The following steps are completely optional. In fact, if you are exploring CAD property mapping for the first time, you should probably skip this step to keep things simple. But for a production environment, you might need to perform steps similar to these to configure the Cost Guide for your users. See *Managing dialog views* for general information.



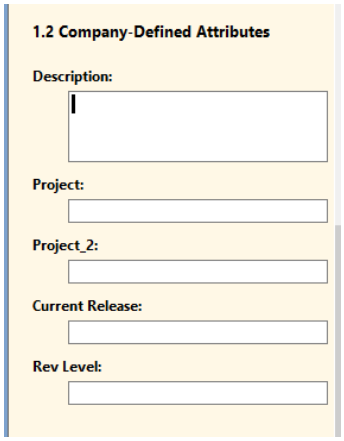
7 Click **Dialog Views** in the left pane.

8 Click the **Search** pull-down menu and click **Cost Guide**.

9 Use the left and right arrows to move entries between the Available Fields: and Displayed Fields:. Use the up and down arrows to change the relative position of items in the Cost Guide. For example, if you do not use the default aPriori fields such as Product Line or CAD Version, you can move them out of the Displayed Fields column to make more room for your new mappings. You can move the new UDAs up

and down if necessary to reflect a logical workflow. Note that you have two similarly-named entries: "Project" is the aPriori system-defined attribute which cannot be mapped, and "Project 2" is the display name of the Project UDA you created in the previous step. We are displaying both to show that a UDA can have the same name as a system-defined attribute but will not conflict with it.

10 When done, click the **Publish Changes** button () in the toolbar. After completing these steps, the Company Attributes section of the **Cost Guide** should now look like the following screenshot (you need to have a part open to display the **Cost Guide**):



Creating an XML Mapping File

Once you have identified the CAD properties that you want to map and have identified or created the aPriori attributes that you want to map them to, you must create the XML mapping file to define these pairings.

Notes:

It is critical that you use an XML-cognizant editor such as Notepad++ that can help you catch syntax errors while editing this file. You can also try to display the edited file in Microsoft Internet Explorer to catch typos and errors prior to using the file. See *Troubleshooting* for more information.

Because installations differ so much from company to company, the following screenshot is only meant to give you an idea of how a valid mapping file might look for this example. We do not provide templates or copies of examples.

1 Create an XML mapping file in an editor that recognizes XML.

2 The following screenshot provides an idea of how these mappings might look:

```
<?xml version="1.0" encoding="utf-8"?>
<database>
        <!-- ordered list of mappings with filters (at run-time
first mapping with matching or null filter will be selected).
        -->
    <mappings>
      <cadPropertyMapping>
            <!-- may specify multiple modelers and model type -->
        <modelFilter modelerTypes="PROE" modelType=" PART">
        </modelFilter>
      <mapping>
            <!-- source CAD property-->
        <source name="ITEM_DESCRIPTION"/>
            <!-- target aP system property -->
        <target name="Description" type="system" />
      </mapping>
      <mapping>
            <!-- source CAD property-->
        <source name="MATERIAL"/>
            <!-- target aP system property -->
        <target name="Material" type="system" />
      </mapping>
      <mapping>
            <source name="REV_LEVEL"/>
                <!-- target aP UDA -->
            <target name="REV_LEVEL" type="uda" />
      </mapping>
      <!-- Alternative mapping using system attribute-->
      <!-- mapping-->
            <!-- source name="REVISION"/-->
                <!-- target aP system Property -->
            <!--target name="REVISION" type="system" /-->
      <!--/mapping-->
      <mapping>
            <source name="CURRENT_REL_NUMBER"/>
                <!-- target aP UDA -->
            <target name="Current_Rel" type="uda" />
      </mapping>
      <mapping>
            <source name="PROJECT"/>
                <!-- target aP UDA -->
          <target name="PROJECT" type="uda" />
      </mapping>
    </cadPropertyMapping>
  </mappings>
</database>
```

3 Save the XML file when done and take note of the folder where you save it.

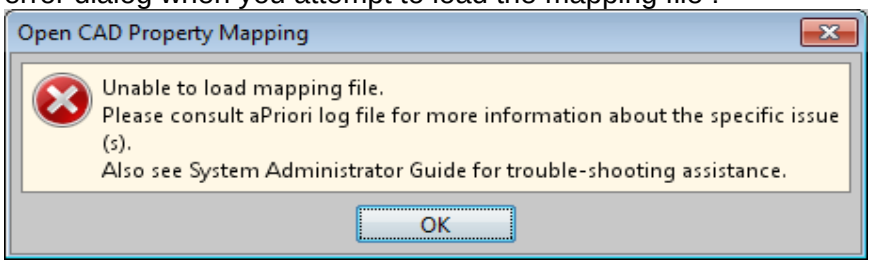
4 IMPORTANT: Ensure that the file contains valid XML with no errors.

5 Load the mapping file: In the aPriori client, click **Tools > System Admin Toolset**.

6 Click the **System Administrator** button () on the resulting window.

7 In the System Administrator window, click **File > Import > CAD Property Mapping File .**

Note: If there is a problem with your XML mapping file, you will see the following error dialog when you attempt to load the mapping file :



Proceed to the *Troubleshooting* section below if you need help tracking down the problem.

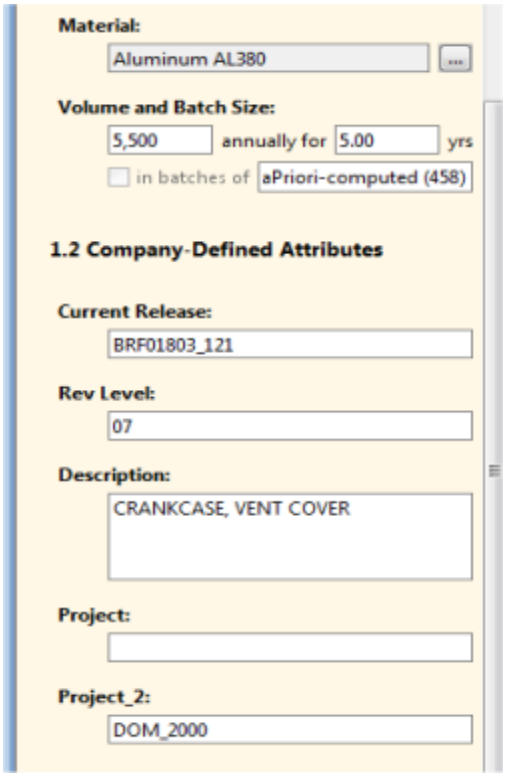
The mapping file is loaded into the aPriori database and will now be used whenever you read or update GCDs from the CAD file and will map the CAD properties to the aPriori attributes.

Testing the Mapping

Once you have saved the XML file, check to see if the mappings appear correctly.

1 Start the aPriori client and open the CAD file.

2 Go to the **Cost Guide** and ensure that the attributes are showing up in the "Company-Defined Attributes" section. (The appearance of your Cost Guide will vary depending on what release you are running and how your UDAs and Dialog View are configured.



In this example, note that the Material field has been updated with the CAD material property, and that the PROJECT UDA (with the display name "Project_2" has displayed the CAD PROJECT property correctly, without conflicting with the system Project attribute.

Using CSL Logic in Your Mappings

As of Release 2018 R1 SP1, you can now embed CSL logic into your mapping files.

This means that it is no longer necessary for the value of the CAD property to exactly match an available value of an aPriori field. For example, the CAD field COMMODITY with a value of “Casting” could be mapped to the “Casting – Die” process group in aPriori. This also means that the "Material" restriction in earlier versions is no longer necessary: CAD properties specifying materials no longer need to use terminology that is 100% the same as aPriori.

The availability of CSL expressions in the mapping file also means that the value of the aPriori field can be driven based on the value of multiple CAD fields as well as just a single CAD field. For example, the aPriori VPE may be chosen based on the values of both the PRODUCTION_REGION and TOOLING_REGION CAD fields.

For a brief introduction to CSL, see the CSL Overview section of the chapter "Working with Cost Model Logic" in the aPriori Professional *Cost Model Workbench Guide*. See also the chapter "Cost Scripting Language Reference" in the same *Guide*.

Basic Examples of CSL Logic

The following snippets show some simple examples of the ability to embed CSL logic into your mapping file. The first example shows a commented-out simple mapping between the Creo "COMMODITY" property and the aPriori Process Group production input. It has been replaced by a CSL expression that checks whether COMMODITY is set to "Casting". If it is, the aPriori Process Group is set to "Casting – Die". If not, the Process Group is set to "Plastic Molding".

```
<!--
    <mapping>
      <source name="COMMODITY"/>
      <target name="Process_Group" type="system" />
    </mapping>
-->
    <mapping>
      <source id="commodity" name="COMMODITY"/>
      <expression cslRule="{'Casting - Die' if source('commodity')=='Casting' 'Plastic Molding' otherwise }" />
      <target name="Process_Group" type="system" />
    </mapping>
```

A more complex CSL expression is used to determine the correct aPriori VPE based on the value of the Creo "PRODUCTION_LOCATION" property:

```
<mapping>
 <source id="productionLocation" name="PRODUCTION_LOCATION"/>
  <expression>
   <cslRule>
     <![CDATA[
{
  with
  {
    productionLocation = source('productionLocation')
    chinaRegion = asList('Beijing','Shanghai','Hong Kong','Shenzhen')
    usaRegion = asList('Boston','New York','Chicago','Atlanta')
  }
  null if (productionLocation == null) _
  'aPriori China' if (productionLocation in chinaRegion) _
  'aPriori USA' if (productionLocation in usaRegion) _
  'aPriori Mexico' otherwise
}
        ]]>
    </cslRule>
  </expression>

  <!-- target aP UDA -->
  <target name="VPE" type="system" />
</mapping>
```

CSL cannot directly access the value of named properties, so the <source> tag also includes an id attribute for CSL to use. The PRODUCTION_LOCATION property may be one of several cities, so the CSL code evaluates whether the city is located in the United States or China and picks the correct VPE for that region. As a back-up, if neither region is recognized, the "aPriori Mexico" VPE is chosen.

Advanced CSL Logic Examples

Using CSL maps

The following mapping determines the appropriate aPriori material name (such as ‘`Aluminum, Stock, ANSI 5052`’) for a variety of possible CAD property values (for example, ‘`ALUMINUM_5052`’). The CAD property name is ‘`MATERIAL_NAME`’ and the aPriori property name is ‘`Material`’:

```
<mapping>
  <source id="material" name="MATERIAL_NAME"/>
  <expression>
      <cslRule>
      <![CDATA[
{
  with
  {
    materialMap = asMap('ALUMINUM_5052', 'Aluminum, Stock, ANSI 5052', _
                        'ALUMINUM_5083', 'Aluminum, Stock, ANSI 5083', _
                        'ALUMINUM_5119', 'Aluminum, Stock, ANSI 5119', _
                        'COPPER_C27200', 'Copper, Stock, UNS C27200', _
                        'COPPER_C28000', 'Copper, Stock, UNS C28000', _
                        'COPPER_C51900', 'Copper, Stock, UNS C51900', _
                        'STEEL_1008', 'Steel, Hot Worked, AISI 1008', _
                        'STEEL_1010', 'Steel, Hot Worked, AISI 1010', _
                        'STEEL_1012', 'Steel, Hot Worked, AISI 1012')
    materialSpec = source('material')
  }
  materialMap[materialSpec]
}
      ]]>
      </cslRule>
  </expression>
  <target name="Material" type="system"/>
</mapping>
```

The built-in CSL function asMap takes as arguments an alternating sequence of keys and values:

```
myMap = asMap(key-1, value-1, key-2, value-2, …, key-n, value-n)
```

You can use an expression of the following form in order to retrieve the value associated with a given key:

```
myMap[key]
```

Here, the aPriori property value is specified by `materialMap[materialSpec]`.

For more information, see the section on Map Functions in the aPriori Professional *Cost Model Workbench Guide*.

Using CSL String Manipulation Functions

The following example provides an alternative method of mapping property values that specify material names (see *Using CSL maps*). This example assumes that the CAD property values have the form shown in the previous example; that is, it assumes that CAD property values consist of either '`ALUMINUM`', '`STEEL`', or, '`COPPER`', followed by an underscore, followed by a standard designation (such as the ANSI designation '`5052`'). This example implements the following mapping:

- '`ALUMINUM_<designation>`' is mapped to '`Aluminum, Stock, ANSI <designation>`'.
- '`STEEL_<designation>`' is mapped to '`Steel, Hot Worked, AISI <designation>`'.
- '`COPPER_<designation>`' is mapped to '`Copper, Stock, UNS <designation>`'.

```
    <mapping>
      <source id="material" name="MATERIAL_NAME"/>
      <expression>
          <cslRule>
          <![CDATA[
{
  with
  {
    standardMap = asMap('ALUMINUM', 'ANSI', _
                        'STEEL', 'AISI', _
                        'COPPER', 'UNS')
    formMap = asMap('ALUMINUM', 'Stock', _
                    'STEEL', 'Hot Worked', _
                    'COPPER', 'Stock')
    materialSpec = source('material')
    materialType = prefix(materialSpec, '_')
    materialCode = suffix(materialSpec, '_')
  }
  msg(mid(materialType, 1, 2), downCase(mid(materialType, 2)), ', ', _
    formMap[materialType], ', ', standardMap[materialType], ' ', materialCode)
}
          ]]>
          </cslRule>
      </expression>
      <target name="Material" type="system"/>
    </mapping>
```

This mapping uses string manipulation functions in order to do both the following:

- Parse the CAD property value into the material type name (`materialType`) and the material standard designation (`materialCode`):
  - `prefix(materialSpec, '_')`: returns the portion of the CAD property value that is before the underscore.
  - `suffix(materialSpec, '_')`: returns the portion of the CAD property value that is after the underscore.
- Convert the material type name from all uppercase to lower case (except for the initial letter):
  - `mid(materialType, 1, 2)`: returns the first character of the material type name. In general, `mid(string, i, j)` returns the portion of `string` that begins with the i^{th} character of `string` and ends with the character immediately preceding the j^{th} character of `string`.
  - `mid(materialType, 2)`: returns the trailing characters of the material type name, that is, all the characters except the first one. In general, `mid(string, i)` returns the portion of `string` that begins with the i^{th} character of `string` and ends with the last character of `string`.
  - `downCase(…)`: returns the result of converting to lower case the trailing characters of the material type name.

The aPriori property value is returned by the predefined CSL function `msg`, which takes a variable number of string arguments, and returns the string that results from concatenating the arguments.

For more information, see the section on String Functions in the *aPriori Professional Cost Model Workbench Guide*.

Using regular expressions in CSL

The following example, like the previous one, illustrates mapping property values that specify material names. And like the previous example, this one identifies a material type name (for aluminum, steel, or copper) and a standard designation (such as the ANSI designation '`5052`') within the value of the CAD property (see *Using CSL String Manipulation Functions*).

But instead of assuming that the property value has a fixed format (type-name>_<designation>), this example handles a variety of formats. It assumes only the following regarding the format:

- Material type name and material designation occur somewhere in the CAD property value.
- For aluminum and steel, the material designation consists of 4 digits (and there is no 4-digit sequence that is *not* the designation).
- For copper, the material designation consists of 'C' followed by 5 digits (and there is no such character sequence that is *not* the designation).

```
<mapping>
  <source id="material" name="MATERIAL_NAME"/>
  <expression>
    <cslRule>
    <![CDATA[
{
  with
  {
    standardMap = asMap('Aluminum', 'ANSI', _
                        'Steel', 'AISI', _
                        'Copper', 'UNS')
    formMap = asMap('Aluminum', 'Stock', _
                    'Steel', 'Hot Worked', _
                    'Copper', 'Stock')
    materialSpec = source('material')
    materialType = {'Aluminum' if (materialSpec like '%[Aa][Ll][Uu][Mm][Ii][Nn][Uu][Mm]%') _
                    'Steel' if (materialSpec like '%[Ss][Tt][Ee][Ee][Ll]%') _
                    'Copper' if (materialSpec like '%[Cc][Oo][Pp][Pp][Ee][Rr]%') _
                    'Aluminum' otherwise}
    materialCode = {listFirst(searchString(materialSpec, 'C[0-9][0-9][0-9][0-9][0-9]')) _
                        if materialType == 'Copper'
                    listFirst(searchString(materialSpec, '[0-9][0-9][0-9][0-9]')) otherwise}
  }
  msg(materialType , ', ', formMap[materialType], ', ', standardMap[materialType], ' ', materialCode)
}
    ]]>
    </cslRule>
  </expression>
  <target name="Material" type="system"/>
</mapping>
```

This example uses CSL `like` expressions in order to identify the material type. A `like` expression returns `true` if the string specified by the left operand matches the pattern specified by the right operand. Patterns are specified by regular expression strings – see the section on Like Expressions in the *aPriori Professional Cost Model Workbench Guide*. Here, the pattern for aluminum is specified by the following string:

```
'%[Aa][Ll][Uu][Mm][Ii][Nn][Uu][Mm]%'
```

This pattern matches any sequence of 0 or more characters, followed by ‘aluminum’ in any combination of upper- and lower-case characters, followed by any sequence of 0 or more characters. The patterns for steel and copper are similar.

The example also uses calls to the predefined CSL function `searchString` in order to identify the material standard designation (assigned to `materialCode`). This function returns a list of all portions of a specified string that match a specified regular expression pattern.

For aluminum and steel, the pattern is specified by the following string:

```
'[0-9][0-9][0-9][0-9]'
```

This pattern matches any sequence of 4 digits.

For copper, the pattern is specified by the following string:

```
'C[0-9][0-9][0-9][0-9][0-9]'
```

This pattern matches any character sequence that consists of ‘C’ followed by 5 digits.

Since we assume that the CAD property value has only one substring that matches the specified pattern, the material designation is assumed to be the first (and only) element

of the list returned by `searchString`. The first element is retrieved with the predefined CSL function `listFirst`.

For more information, see the section on String Functions in the aPriori Professional *Cost Model Workbench Guide*.

Using CSL `foreach` Expressions

The following example illustrates a method for determining the aPriori VPE corresponding to a CAD property that specifies a city. It is similar to an earlier example (see *Basic Examples of CSL Logic*), but this method uses a foreach expression instead of a conditional (if) expression:

```
      <mapping>
        <source id="productionLocation" name="PRODUCTION_LOCATION"/>
        <expression>
          <cslRule>
            <![CDATA[
{
  with
  {
    chinaRegion = asList('aPriori China', 'Beijing','Shanghai','Hong Kong','Shenzhen')
    usaRegion = asList('aPriori USA', 'Boston','New York','Chicago','Atlanta')
    germanyRegion = asList('aPriori Germany', 'Wolfsburg', 'Stuttgart', 'Munich')
    regions = asList(chinaRegion, usaRegion, germanyRegion)
    productionLocation = source('productionLocation')
  }
  foreach (region : regions) getFirst(r) {
    r = { listFirst(region) if (productionLocation in region) null otherwise }
  }
}
            ]]>
          </cslRule>
        </expression>
        <target name="VPE" type="system" />
      </mapping>
```

Here, `chinaRegion` is a list whose first element is the name of the China VPE ('aPriori China'), and whose remaining elements are the names of the cities associated with that VPE. The lists `usaRegion` and `germanyRegion` have the same form. The formula `regions` is a list of lists; each of its elements corresponds to a region.

The `foreach` expression builds a result set by considering each region in turn:

```
foreach (region : regions) getFirst(r) {
    r = { listFirst(region) if (productionLocation in region) null otherwise }
    }
```

If the CAD property value specifies a city that is in the region under consideration, the `foreach` expression adds to the result set the region's associated VPE (the region list's first element, obtained with the predefined function `listFirst`); otherwise the `foreach` expression adds a null value to the result set:

```
foreach (region : regions) getFirst(r) {
    r = { listFirst(region) if (productionLocation in region) null otherwise }
    }
```

The `foreach` expression uses the reduction function `getFirst` to return the first (and, presumably, only) non-null element of the result set:

```
foreach (region : regions) getFirst(r) {
    r = { listFirst(region) if (productionLocation in region) null otherwise }
    }
```

To modify this mapping so that it handles a new, additional region, you would add a list for the region by using a formula that has the following form:

```
<new-region> = asList(<vpe-name>, <city-name-1>, …, <city-name-n>)
```

You would also modify the `regions` formula to add the new region:

```
regions = asList(chinaRegion, usaRegion, germanyRegion,
   <new-region>)
```

For more information, see the section on foreach Expressions in the *aPriori Professional Cost Model Workbench Guide*.

Troubleshooting

The most common problem when mapping properties is malformed XML. You should always use an editor that recognizes XML code and which flags syntax errors. You can also quickly check that an XML file is valid by trying to open it with Microsoft Internet Explorer (MSIE). If it has an error, it will not display at all.

Errors can also be caused by defining sources or targets that do not exist, or which do not match. Note that CAD properties and aPriori attributes are case-sensitive: if your XML syntax looks correct, and the properties and attributes that you are trying to map all exist, make sure that your spelling in the XML file exactly matches the case the names.

If you encounter a mapping problem and cannot track it down, examine the aPriori log file (apriori.log). Most problems will result in an error message that can be found in this file. You can access this log from the aPriori client by clicking **Help > Explore Logs Directory** and then double-clicking apriori.log.

9 Adjusting Heap Memory

This chapter tells you how to configure heap memory for aPriori Professional.

This chapter includes the following topics:

- Types of memory
- Reasons for increasing heap
- Types of aPriori heap
- Adjusting aPriori heap size

Types of memory

When you are configuring your computer(s) for aPriori software, you must consider two kinds of memory:

- RAM – the total hardware memory installed in your machine
- Heap – portion of that memory which is allocated aPriori software.

Reasons for increasing heap

Since heap size determines how much memory is allocated exclusively to aPriori software, it can have a significant impact on how aPriori performs. If you work with a large number of very complex CAD parts, you will probably benefit from increased RAM memory and an increased heap size setting.

To determine how much RAM you should install and how large your heap size settings should be, please refer to “Basic Requirements” in the latest aPriori *System Requirements* document.

Types of aPriori heap

There are different heap size requirements and different procedures for adjusting heap size for:

- aPriori Professional in Creo direct integration mode
- aPriori Professional in standalone mode
- aPriori Professional bulk costing
- Cost Insight Admin and Cost Insight Report

This chapter covers aPriori Professional, in both Creo direct integration mode and in standalone mode. For adjusting heap for bulk costing procedures, see the aPriori *System Requirements* document. For Cost Insight Admin and Cost Insight Report, see the *Enterprise Platform Installation Guide* and the *Cost Insight Administration Guide*.

Adjusting aPriori heap size

Note: Adjusting heap size is not particularly difficult, but if you get it wrong (such as allocating more memory to heap than actually exists on your machine), you can disable your computer. If you have the slightest doubt about performing these procedures, please contact either your internal IT team, or aPriori Customer Support.

First, determine how big your heap size setting should be by referring to the *System Requirements* document. By default, aPriori currently ships with heap size set to 4 GB

(4096 MB). For the purposes of these examples, we'll assume that you want to double that setting to 8 GB (8192 MB) but use the number that is correct for your situation.

Then use the procedure below that is correct for your environment to change the setting. Note that you can do this either by editing the aPriori start-up file `runStandaloneApriori.cmd`, or by setting an environment variable `APRIORI_JAVA_OPTS`. Make sure to restart aPriori for this change to take effect.

For Standalone aPriori: Start-up File

1. Navigate to the /bin folder in your aPriori installation folder.
2. Open runStandaloneApriori.cmd in your favorite text editor (for example, Notepad++).
3. Find the line that starts:

   `set APRIORI_JAVA_OPTS=-Xmx4096m…`
4. Change to "4096" (or whatever the correct value is for your installation. Only change the numbers between the Xmx and the m.
5. Save the file and restart aPriori.

For Standalone aPriori: Environment Variable

You can alternatively change the heap setting by using an environment variable.

1. Create a system environment variable named "APRIORI_JAVA_OPTS".
2. Set the value of this new environment variable to "-Xmx8192m" (or whatever the correct value is for your installation).
3. Save/apply changes and restart aPriori.

For Creo Direct Integration: Start-up File

You should do the following with the assistance of your internal IT team:

1. Open the Creo start-up script in your favorite text editor.
2. Add the following line:

   Set APRIORI_MAX_MEMORY=8192m (or whatever the correct value is for your installation).
3. Restart Creo.

For Creo Direct Integration: Environment Variable

You should do the following with the assistance of your internal IT team:

1. Create a system environment variable named "APRIORI_MAX_MEMORY”.
2. Set the value of this new environment variable to “8192” (or whatever the correct value is for your installation).
3. Save/apply changes and restart Creo.

10 Properties Files

aPriori maintains properties files where users, administrators, and aPriori services & support personnel can configure certain aspects of system behavior. The properties in this chapter are those which customers may wish to use.

This chapter includes the following topics:

- Overview of Properties Files
- apriori.properties
- bulkLoad.properties
- plugin.properties
- Other properties files

Overview of Properties Files

aPriori provides a number of files in which properties can be set to control the behavior of various parts of the system. The most important of these include:

- apriori.properties – This is the primary properties file and determines the behavior for all users in an installation (unless overridden at a local level). See *apriori.properties* below.
- apriori.user.properties – This file is similar to apriori.properties, but controls only the settings for a specific user. See *apriori.user.properties* below.
- bulkLoad,properties – This file specifies the predefined bulk costing inputs required by the bulkLoad.cmd. See *bulkLoad.properties* below.
- plugin,properties – These files exist for each of the aPriori plugins that are in use at your installation. See *plugin.properties* below.

Some of the settings in these files are intended for modification by customers, but many settings exist for adjustment only by aPriori Support or Services personnel. If you have the slightest question about whether you should modify one of these settings, contact aPriori Support first.

apriori.properties

The apriori.properties file is created fresh during every installation, and resides at the top level of the installation directory:

`<apriori_install_dir>\<apriori_version>\install\ apriori.properties`

The entries in the table below are the only properties that customers should change. You should only add or modify these if you understand exactly what they do. If you have any questions, please consult with aPriori Professional Services or Support.

Property	Description
apriori.doc.cache.in.user.home **apriori.enable.doc.cache.cleaner** **apriori.doc.cache.dir**	`[true|false]` The first two entries take true/false arguments and are set automatically during the installation and setup process. The third takes a path argument for locating the cache in a custom directory. These properties generally should not be modified except under the direction of aPriori Professional Services or Support. For an explanation of the document cache, see the aPriori *Installation Guide*. **Note:** The Bulk Costing properties file (`install\ext\analysis-purchasing-module\plugin.properties`) contains a setting to control cache cleaning upon

Property	Description

	execution of the Bulk Loader:
	`launcher.set.apriori.enable.doc.cache.cleaner=false`
	This cleaner addresses the VPE and cost model caches but does not touch GCD/images. aPriori provides a separate script to clean the GCD/images cache:
	`install\bin\cleanDocCache.cmd`

`apriori.cad.flattening.cache.timeout.seconds`
`apriori.cad.flattening.cache.folder`
`fbc.debug.geom.save.bse.files` (for debugging purposes only)
	The flattening cache applies only to the flattening capabilities of the Sheet Metal Process Group. See the aPriori *Cost Model Guide* for more information about this capability.
	The timeout value determines how long cached flattened output is kept. The default value is 4 days.
	The folder path determines where the files are cached. The default is:
	`C:\Users\<username>\AppData\Local\aPriori\<version>\tmp\flatteningCache`
	Note that the third property is for debugging use only and should only be changed in consultation with aPriori Support.
	Can be set to `true` or `false`. Default = `false`. If set to `true`, intermediate input files that aPriori sends to the flattening subsystem are dumped to the user's cache location, e.g.
	Files will be saved in
	`C:\Users\<username>\AppData\Local\aPriori\<version>\tmp\BSE\<partName>`
	`<partName>` is the filename of the component being costed.
	There will be an XML and a SAT file for input, and another XML/SAT pair for output. The output files will be in a `bseOut` subfolder.
	The output XML contains all error codes reported by the flattening subsystem.

`apriori.display.locale`	`de_DE = German`
	`fr_FR = French`
	`en_US = American English` (default)
	Specify the language and its locale for the aPriori GUI. For a full description of these

Property	Description
	options, see "Selecting a Language for the aPriori Interface" in the aPriori User Guide.
apriori.max.rows.to.expand	Specifies the number of rows that are expanded in the Assembly Details tab. Default is 1000 rows

apriori.enable.surface.models
 	
apriori.enable.surface.models.sheetmetal
	`[true|false] Default=false`
	Allows you to cost CAD models that are either "surface-only" (having no solid geometry), or hybrids that have some combination of solid and surface-only features. Applies ONLY to User Guided process groups.
	
	`[true|false] Default=true`
	Allows you to cost CAD models that are either "surface-only" (having no solid geometry), or hybrids that have some combination of solid and surface-only features. Applies ONLY to Sheetmetal and Sheetmetal – Transfer Die process groups.
	
	NOTE: For details about the interaction between these two similar properties, see the section following this table, "*The enable.surface.model Properties*".

apriori.keep.free.bodies **apriori.free.bodies.preserve.CAD** **apriori.free.bodies.ignore.missing.component**	These properties specify how aPriori should handle CAD files containing more than one distinct solid body. Siemens NX files in particular may contain a combination of sub-components and other solid bodies due to specific NX modeling techniques. All of these properties take `[true|false]`values, and the default is `false`. `apriori.keep.free.bodies` – The behavior of this property depends on whether or not the CAD file is an NX assembly file. When set to true: ■ Any CAD part file which contains multiple distinct (detached) solid bodies causes aPriori to interpret the file as an assembly with a subcomponent for each solid body. (For all CAD systems, a part file can

Property	Description
	contain detached solids, e.g. two extruded profiles which are not attached. Also note an NX file is considered a "part" if there is only one component in the Assembly Navigator pane AND an icon of a single cube appears beside it.) ■ An NX assembly file causes aPriori to interpret the file as an assembly scenario with a subcomponent for each solid body and NX subcomponent. When set to false, only the largest solid body in the part file will be extracted as a part scenario. In an NX assembly file, only the subcomponents will be extracted, not the solid bodies. (When aPriori creates additional subcomponents from the free bodies, it generates names for them following the convention <*filename*>-AP_PART-1, <*filename*>-AP_PART-2, etc.) **Note:** `apriori.keep.free.bodies` does not apply to Creo or CATIA assemblies, as it is not possible to have a "free body" protrusion existing in an assembly in those file formats. `apriori.free.bodies.preserve.CAD` – Applies only to NX assembly files when `apriori.keep.free.bodies` is set to "true." When set to true, all solid bodies and subcomponents in the NX file are imported. When set to false, aPriori excludes subcomponents that significantly overlap solid bodies as the solid bodies typically represent a finished part derived from the subcomponent. `apriori.free.bodies.ignore.missing.component` – Applies only to NX files with a rare error: When set to "true" NX assembly files containing exactly one free body and one missing component will be interpreted as a part with the free body's geometry. A "missing component" is one for which the corresponding CAD file cannot be found in the expected location. **NOTE:** These properties apply to both interactive costing and Bulk Costing and Analysis. However, the Bulk Loader also recognizes two additional properties. See *bulkLoad.properties* below as well as the "Bulk Costing and Analysis" chapter in the aPriori Professional *User Guide* for more information.

Property	Description
	As of Release 19.1, aPriori Professional also provides a Multi-Body Options dialog for setting these values. See “Multi-Body Options” in the “Changing User Preferences” section of the aPriori Professional *User Guide*.
apriori.viewer.projection=Perspective	`[Orthographic | Perspective ]` Changes the Component Viewer display from Orthographic (default) to Perspective.
apriori.max.image.size.mb	Use this property to increase the size of component images that are permitted to be saved to the database. For example, if you open a large part but its image does not appear in the UI, try increasing this value. Default value is “`10`”. Recommended increased value is “`50`”.
apriori.interop.dlls.dir	*<path_to_DLL_directory>* By default, aPriori plugins use DLLs from Spatial Interop as specified by the installer. For example: *(<install_dir>*`\WIN64\catia\NT_VC11_64_DLL\code\bin`*)* However, some plugins might require different versions of these DLLs. This property allows you to specify a directory containing DLLs of a different version.
gcds.to.highlight.threshold	`Default setting = 2000` Implemented for efficient handling of large parts with thousands of GCDs. aPriori will only automatically highlight and select GCDs when the GCD category node has fewer GCDs than this setting. If this threshold is exceeded, NO automatic highlighting occurs, in order to prevent unacceptable delays in the GUI.

The enable.surface.model Properties

As you may have noticed in the above table, there are two very similar properties:

- **apriori.enable.surface.models.sheetmetal**
- **apriori.enable.surface.models**

`apriori.enable.surface.models.sheetmetal` was introduced in aPriori Professional Release 2018 R1, to force virtual thickening of surface models in Sheet Metal process groups. It is set to “true” by default.

`apriori.enable.surface.models` is an older property that has been available for several releases and which has been retained to preserve behavior in non-Sheet Metal process groups. It is set to “false” by default.

For Sheet Metal GCD extraction, `apriori.enable.surface.models.sheetmetal` supersedes `apriori.enable.surface.models`.

The following table shows the expected behavior for the various combinations of these properties:

apriori.enable. surface.models. sheetmetal=	apriori.enable. surface.models=	Behavior in Sheetmetal	Behavior in other process groups	Behavior in User Guided
true(default)	false (default)	Part displayed. GCDs extracted.	Part displayed. GCD extraction **error.**	Part displayed. No GCD extraction error.
true (default)	true	Part displayed. GCDs extracted.	Part displayed. GCD extraction **error**.	Part displayed. No GCD extraction error.
false	false (default)	Part **not displayed**. GCD extraction **error** (part never opened).	Part **not displayed**. GCD extraction **error** (part never opened).	Part **not displayed**. GCD extraction **error** (part never opened).
false	true	Part displayed. GCD extraction **error**.	Part displayed. GCD extraction **error**.	Part displayed. No GCD extraction error

So how do these property settings translate to what you actually see on your screen? Here are two common settings and how they affect costing in different process groups.

EXAMPLE 1:

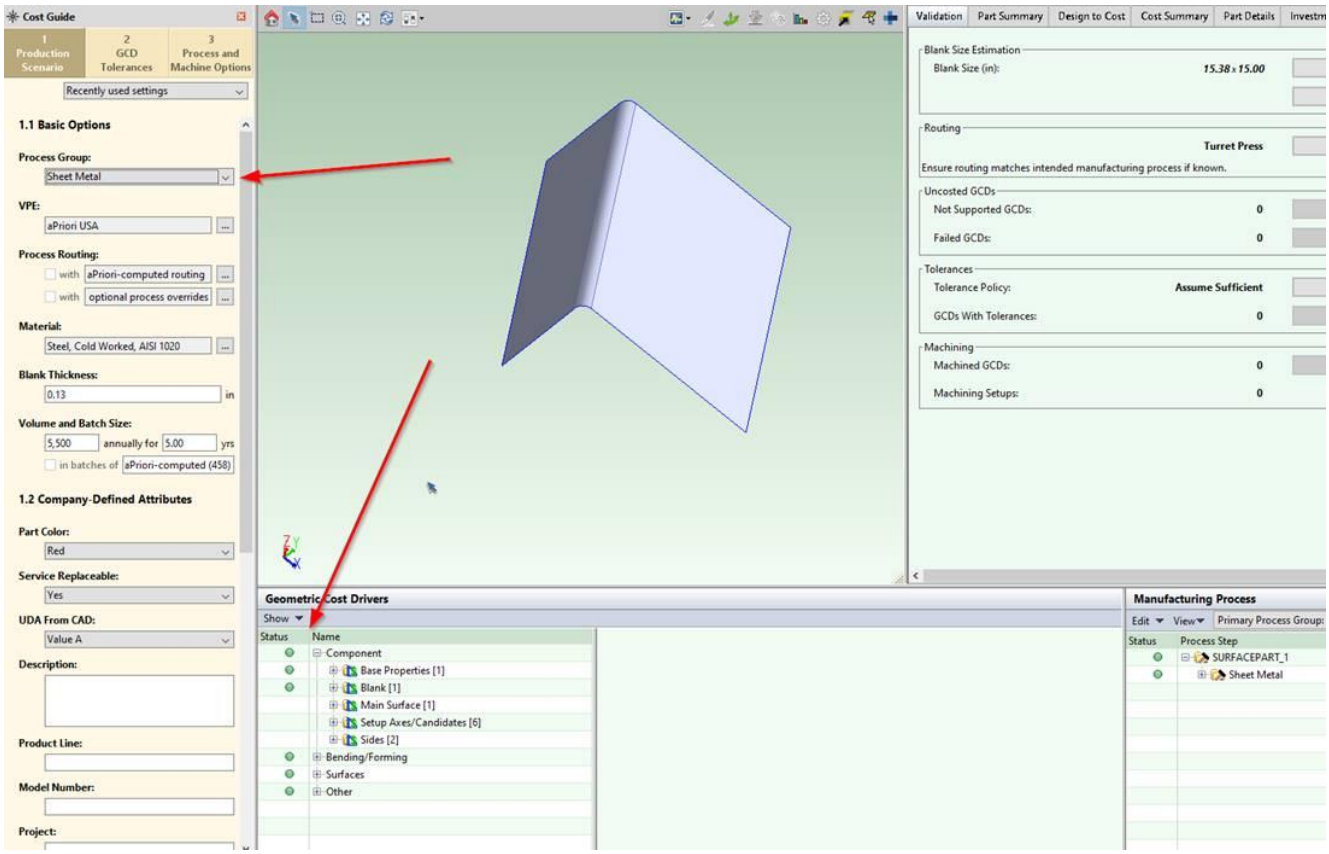
apriori.enable.surface.models=**false**

apriori.enable.surface.models.sheetmetal=**true**

Process Group = Sheet Metal

Result:

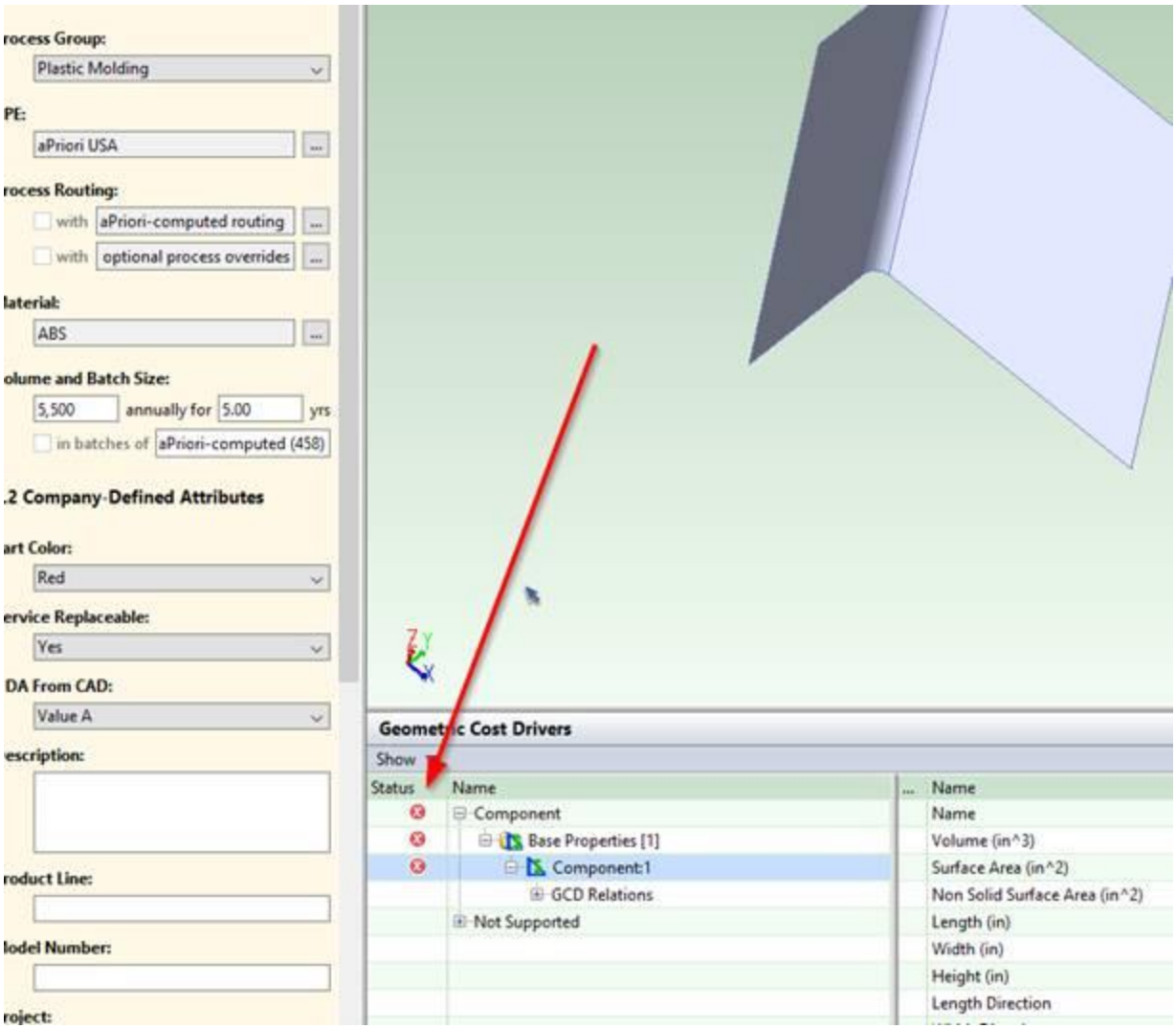
- Part is displayed
- Able to extract GCDs like a solid model (No GCD extraction errors)



Process Group = Plastic Molding

Result:

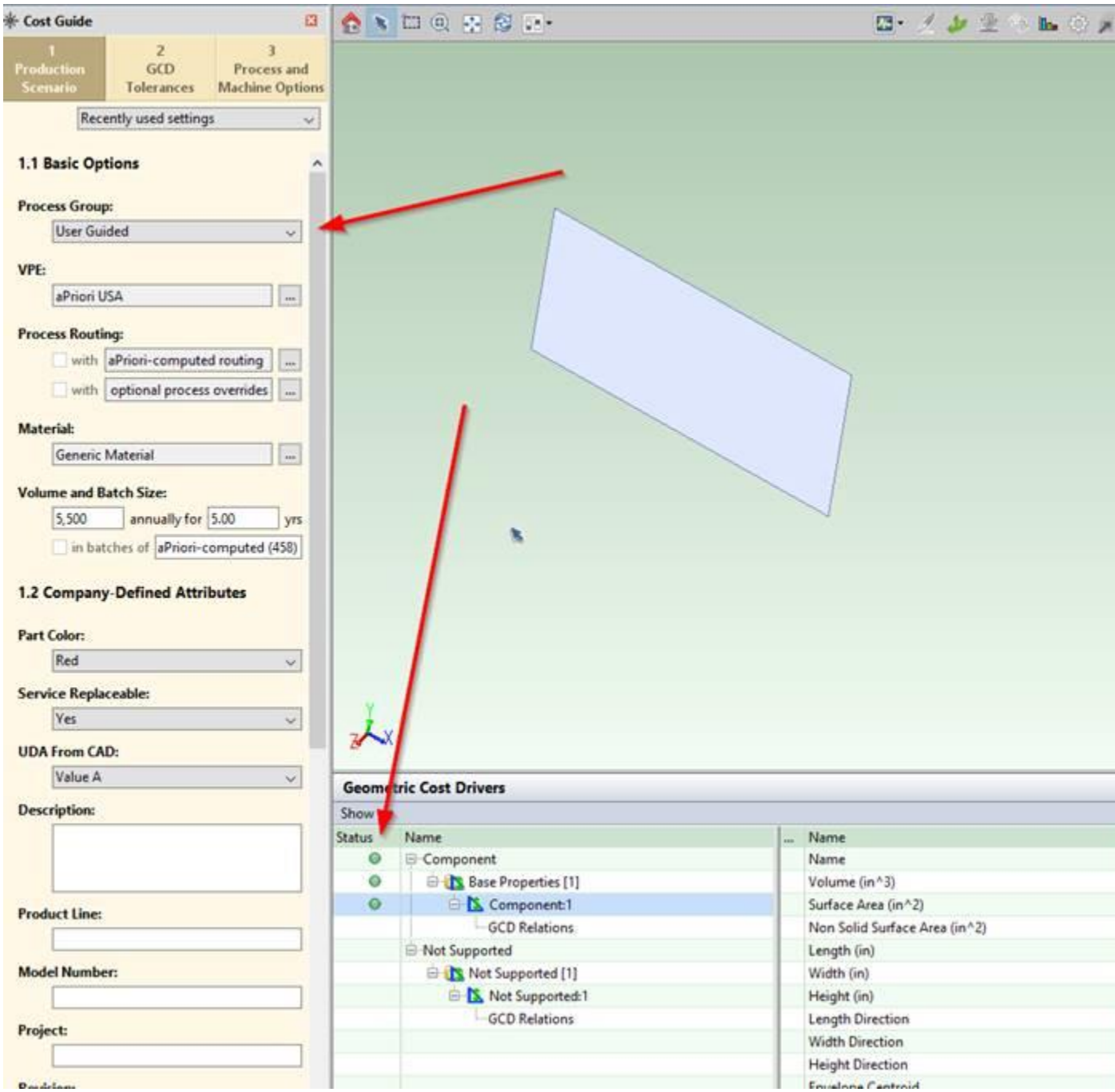
- Part is displayed on screen, but when first trying to cost, error (see below) is displayed: “This model has no solid geometry”.
- No GCDs extracted because there is no solid geometry
- GCD extraction “errors”, saying surfaces are not supported, see red dots with X in GCD panel



Process Group = User Guided

Result:

- Part is displayed on screen, when first trying to cost, NO error is displayed.
- No GCDs extracted because there is no solid geometry.
- No GCD extraction error (note all green dots for GCDs below) but that is because there ARE no GCDs.


Cost Guide
1 Production Scenario
2 GCD Tolerances
3 Process and Machine Options
Recently used settings
1.1 Basic Options
Process Group:
User Guided
VPE:
aPriori USA
Process Routing:
with aPriori-computed routing
with optional process overrides
Material:
Generic Material
Volume and Batch Size:
5,500 annually for 5.00 yrs
in batches of aPriori-computed (458)
1.2 Company-Defined Attributes
Part Color:
Red
Service Replaceable:
Yes
UDA From CAD:
Value A
Description:
Product Line:
Model Number:
Project:
Geometric Cost Drivers
Show
Status
Name
Component
Base Properties [1]
Component:1
GCD Relations
Not Supported
Not Supported [1]
Not Supported:1
GCD Relations
Name
Name
Volume (in^3)
Surface Area (in^2)
Non Solid Surface Area (in^2)
Length (in)
Width (in)
Height (in)
Length Direction
Width Direction
Height Direction

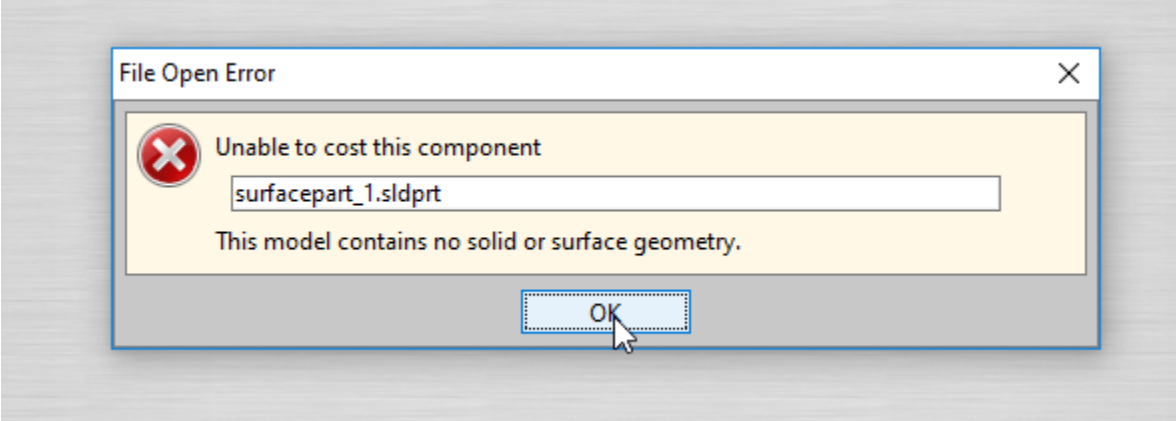
EXAMPLE 2:

apriori.enable.surface.models=**false**

apriori.enable.surface.models.sheetmetal=**false**

Process Group: ALL

Result:



apriori.user.properties

This file typically resides in:

`C:\Users\<username>\AppData\Local\aPriori\<apriori_version>\`

Unlike the apriori.properties file above which applies to all users, this file applies only to the local user. It performs two purposes:

- It stores preference and display information so that certain display settings are remembered every time you log in.
- It allows you to set properties that apply only to your client, not to the entire installation.

The following properties are usually set only in `apriori.user.properties`:

cost.table.decimal.places

This property allows you to display more decimal places in various aPriori views including the **Cost Summary** tab, **Part Details** and **Assembly Details** tab, **Investment** tab, **Rollup Summary** and **Comparison Summary** views.

`cost.table.decimal.places = <n>`

By default, aPriori displays two significant digits after the decimal point for numeric values in these views.

Note that certain dialogs will NOT be affected by this option and will always be displayed with two decimal places, including the **Cost Ticker** and the **Materials Selection** dialog.

apriori.file.open.dialog.use.static.solidworks.icons

This property enables SolidView users to speed up the display of the aPriori **Open CAD File** window when opening folders containing many SolidView files.

The issue is caused by the time it takes to generate thumbnail preview images for each part. The property turns off this behavior and instead displays type-specific static icons rather than thumbnail preview images. To implement this fix, add the following line:

```
apriori.file.open.dialog.use.static.solidworks.icons=true
```

to the apriori.user.properties file.

Controlling Component Color Using Properties

The aPriori component viewer uses yellow or magenta (purple) to highlight selected surface and volumetric GCDs. Since it is difficult for users to see highlighting against similarly-colored parts, aPriori displays such parts in gray instead of their original color.

You can configure the thresholds for which part colors are considered “too similar” to aPriori highlighting colors, in either apriori.properties or user.apriori.properties.

Use the following two properties to specify CAD model colors that should be displayed as gray within aPriori:

- `apriori.ui.prohibited.part.colors` (defaults to "yellow,magenta")
- `apriori.ui.prohibited.asm.colors` (defaults to "yellow")
  - The format of these properties is "color1,color2,color3" where each comma-separated value is either a JAVA-compatible color string or RGB hex value 0xRRGGBB. For reference, Java color strings (case sensitive) are

    white, WHITE, lightGray,LIGHT_GRAY, gray,GRAY, darkGray, DARK_GRAY,black, BLACK, red, RED, pink,PINK, orange, ORANGE, yellow, YELLOW, green, GREEN, magenta, MAGENTA, cyan, CYAN, blue, BLUE

Note that parts with a “similar” color to a designated color also will be displayed as gray. For example, light yellow and dark yellow parts also are switched to gray, as well as bright yellow parts) The degree of similarity is controlled by a third property:

- `apriori.ui.prohibited.color.similarity` (defaults to 5)

  Valid values are 0-100 (percentage) or -1, determining how close part colors needs to match an apriori reserved color to be prohibited.
  - 0 means require perfect match, 100 means any color with even mildest hue component matches, default is 5.
  - Set to -1 to always use part color and never reset to gray.

  In order for a change to this property to take effect for a given part, you must restart aPriori and then extract or re-extract GCDs for the part.

bulkLoad.properties

The bulkLoad.properties file is read when you run the bulk loader from the command line. You can save the file wherever you like so long as you provide the path when you run the command. A sample template file is provided at:

`<apriori_install>\ext\analysis-purchasing-module\command-line\bulkLoadTemplate.properties`

The `bulkLoad.properties` file is documented in detail in the "Bulk Costing and Analysis" chapter of the aPriori *User Guide*.

plugin.properties

aPriori plug-ins are optional software packages that extend the capabilities of your installation. Most plug-ins typically have their own plugin.properties files that determine the behavior of the plug-in. Each plugin.properties file typically has at least one setting to turn the package on ("false") or off ("true"):

```
plugin.disabled=false
```

While some plug-ins include only this one-line, other plug-ins may have several other settings as needed. For example, the bulk costing plug-in (analysis-purchasing-module) has over two dozen additional properties to control its behavior. In-line comments within these file document these settings.

Plug-in property files reside in the following typical location:

`<apriori_install>\ext\<plugin_name>\plugin.properties`

Other properties files

Other properties can be set in additional files, but in general these should only be modified after consulting with aPriori Support or Services personnel. These include:

- `C:\Users\<username>\AppData\Local\aPriori\<apriori_version>\costinapriori<CAD system>.properties`
- `<apriori_install_dir>\<apriori_version>\install\hibernate.properties`
- `<apriori_install_dir>\<apriori_version>\install\install-files\wan-properties\apriori.system.properties`
- `<apriori_install_dir>\<apriori_version>\install\log4j.properties`
- `<apriori_install_dir>\<apriori_version>\install\bin\runStanadaloneApriori.cmd` (Although this is not really a properties file, it contains a memory setting that can be changed with upon consultation with aPriori Support or Services.)
- `configuration.properties` (If you install any of the Enterprise Platform modules such as Scenario Synchronization or Cost Insight Report, you will need to set several properties in this file. See the Server documentation for more information.)

aPriori Technologies, Inc.

300 Baker Avenue

Concord, MA 01742