DEVELOPER'S GUIDE

Axway Application Studio

Version 1.4

21 April 2016

This documentation describes the following Axway software:

Axway Application Studio 1.4

Contents

Preface

This guide provides documentation about specialized Axway Application Studio functionality not found in the online Joget Workflow knowledge base.

About Application Studio

Application Studio combines development and operations in one product. It is a rapid application-development environment for building web applications that implement business processes. It is also an application operations environment for business users.

Application Studio is rapid in the sense it uses model-driven, visual development approaches rather than code for much, if not all, of the application development process. It does involve software-development engineering, but lets developers work at a higher level of abstraction than when coding. Business processes are designed using sequences of activities that involve participants, represented as blocks laid out in swim lanes customary to BPM. Forms are created by dragging visual elements from a palette onto the form canvas.

Application Studio is built on Joget Workflow Enterprise Edition, a platform for building enterprise web applications.

Who should use this guide

This guide is for people who use Application Studio to design workflows and applications. This guide presumes you have a knowledge of:

- Your company's business processes and practices
- Your company's hardware, software and IT policies
- The Internet, including use of a browser

Others who may find parts of this guide useful include relationship managers, network or systems administrators, database administrators and other technical users.

Axway 5 Suite reference solutions

Application Studio, a Unified Flow Management product, is integral to all Axway 5 Suite reference solutions. These meld selected Axway products into seamlessly integrated systems:

- Manged File Transfer to securely transfer data in one-to-one, one-to-many and many-to-many scenarios.

- B2B Integration to exchange, transform and process standardized business documents withing an enterprise's B2B community
- Data Flow Integration to provide services for standardizing the exchange of business data with internal and external partners.
- Financial Integration to support data transfers in finance channels such as SWIFT and EBICS and transformations of data in financial protocols.

Your organization might use this product in context of a reference solution. Find details about the product's role in documentation on the Axway Sphere support website at support.axway.com.

Other documentation

Refer to the Application Studio Help Center and the Joget knowledge base for information about installing, getting started and using Application Studio. The URLs are:

Application Studio Help Center

http://app-studio-help-center.squarespace.com/

You also can click Help Center in the user interface to access it.

Joget knowledge base

http://dev.joget.org/community/display/KBv4/Joget+Workflow+v4+Knowledge+Base

Application Studio uses Joget Workflow 4, and this URL opens version 4 of the knowledge base.

Axway online

Go to Axway Sphere at support.axway.com to contact a representative, learn about training programs, or download software, documentation and knowledge-base articles. Check Sphere before installing and at intervals thereafter for the latest product documentation. Sphere is refreshed periodically with updated user documentation.

Sphere is for customers with active Axway support contracts. You need a user name and password to log on.

Accessibility

Axway strives to create accessible products and documentation for users. The following describes the accessibility features of Application Studio and its documentation.

Product accessibility

Application Studio has the following accessibility features.

The Application Studio user interface supports browsers with accessibility features, including keyboard shortcuts. Check the browser user documentation for information.

Documentation accessibility

The product documentation provides the following accessibility features.

- Alternative text is provided for images whenever necessary.
- The PDF documents are tagged to provide a logical reading order.
- The documentation can be used in high-contrast mode.
- There is sufficient contrast between the text and the background color.
- The graphics have the right level of contrast and take into account the way color-blind people perceive colors.

Application Studio overview 1

This topic introduces some high-level concepts and things to consider before developing an application using Application Studio.

Target audience

Application Studio is for application developers keyword with experience building applications using programming languages such as C++ or Java. Although writing code is, in most cases, unnecessary with Application Studio, your experience as a software developer is critical in conceptualizing and documenting how the internal software components relate to each other, organizing how information is stored in your database, your naming conventions, the variables you need to maintain, and a host of other details. Further, you need to create applications and their accompanying documentation with a focus on maintainability.

The development environment

Application Studio is a rapid application development (RAD) platform for building business process applications.

Rapid application development

Application Studio enables RAD instead of more labor-intensive conventional programming languages. Application Studio does not require writing code to build applications, although there may be specific cases where you might wish to extend the built-in functionality, such as adding unique form field validators, providing integration with external systems, and so on.

If you do require custom code, the platform is extensible using plugins written in Java.

RAD means you can develop applications faster than conventional coding. Although the Application Studio RAD platform is powerful, it requires great attention to detail as you conceptualize, track, develop, build and scale substantive applications.

Business process applications

Effective business process applications engage end users directly into participating in business processes. Users participate by initiating and responding to requests, by supplying information as subject matter experts, and making judgments providing approvals, rejections, or requests for clarification at various stages in the development process.

Application Studio allows you to send informational and notification emails and integrate with external systems like Axway's middleware software products as additional steps in the processes.

A business process proceeds from step to step, or possibly to one of several different steps (splits), depending on information supplied by users, responses from external systems that are called, or whether specific requests and submissions are approved or rejected by users.

Multiple steps can proceed in parallel, rejoining a common path (joins) when one or all parallel steps are completed. These business processes are exposed to application users in a web-based business process portal where users log in to participate in the business processes and get visibility into their state.

Model-driven, visual development

Business processes are modeled as workflows using the built-in visual tools that provides graphic representations that support business process management.

Workflows, forms, data lists and the portal layout are built using visual tools.

Operations environment

The same RAD platform also is the execution platform for applications built on it. Application Studio executes the business processes modeled during development, exposing a browser-based web application – in effect, a business process portal presented to application end users. So the same Application Studio executable (or more likely a separate instance of the Application Studio executable, deployed in production) is also the production operations environment for IT personnel.

Applications can be exported from one instance and imported into another. Promotion of an application from the development environment is by means of exporting the application. The export is a zip file comprising two compressed and archived files: an XML application definition file and an XPDL workflow definition file.

Application Studio runs the application by executing the model defined by these two files. In this way, it is model-driven development and execution. The combined development and operations (DevOps) environment is available to users with administrative privileges for developing and deploying applications, and for monitoring their operation.

The business process web portal

Separate from the DevOps environment is the runtime application itself, served up by Application Studio. It is accessed through a URL unique to the application and presented as a business process web portal. The application exposed to end users can be styled and branded to match your corporate messaging. Each instance of Application Studio contains the combined DevOps environment and the business process portal exposed to end users. You can use one or more instances in development:

- One or more for QA and testing
- Instances in pre-production that simulate the production environment
- One instance or multiple clustered instances in production

Tools for application development

Application Studio guides you with user-interface controls and widgets and application behaviors that enable developing the applications you need in ways that enhance the creative process.

The following are visual design tools for building applications.

Workflow designer

Use the Workflow designer to lay out the business process represented as a workflow using the visual tool. Workflows involve:

Participants

Users or systems that initiate a process, other users that have a role in the process, and certain system participants used for sending email or sending web service calls to external systems.

Activities

Performed by participants, activities are actions such as the need to fill out a form or approve information submitted by another user.

Tools

There are tools included with Application Studio, such as for invoking email notifications, that can be added to the workflow, or create custom tools to use for your specific purpose.

Connections between activities

Direct paths and paths through:

- Splits to one of many next activities or to multiple next activities
- Joins converging multiple paths to a common next activity

Swim lanes

Swim lanes, horizontal or vertical, represent the various participants in the workflow diagram. For example, participants can be the initiator, subject matter experts, approvers, and the system itself. In the Workflow Designer, you create a swim lane for each participant, drag-and-drop activities representing each workflow activity into the appropriate swim lane, then connect them either directly or using splits or joins, as needed.

Best practice is to design and test the workflow wireframe. Once the workflow is behaving as expected, proceed to the next step of defining the forms that attach to activities.

Form builder

As you design forms, Application Studio creates the database schema in the background. The schema may affect how the application scales and its ultimate performance. Therefore, it is best to think database schema first, then proceed to design the forms in a way that causes Application Studio to create the desired underlying schema.

Each form you design is associated with a database table you specify. A form consists of fields that you drag-and-drop onto the form at design time. These fields are represented in the database as columns. You can also embed subforms that you create. Subforms may involve columns from different database tables. Additional forms may involve existing or new columns, or both, in an existing table or new columns in a new table.

One extreme is to use a single table for all forms. This practice is inefficient and significantly hinders database performance as you scale up application usage. The other extreme is a separate table per form. That likely won't work because more than one form may involve the same data – a single piece of data should be stored in just one place in the database. Keeping these extremes in mind, you can optimize your schema somewhere between them.

Datalist builder

Datalists are partial-to-complete views of database tables. These can be displayed as the workflow proceeds and as forms are completed and submitted, populating the database.

You can filter a subset of records to display and choose the columns to display. The application end user also can specify additional filter criteria for customizing their view. Lists can be displayed to participants at various points in the business process workflow. Lists can be displayed directly in response to clicking a Userview menu item.

Userview builder

The Userview is the visual framework of the web-based business process portal exposed to users when they access the application. The left column contains menu items. You can choose what types of users see which menu items based on the organization they belong to (such as yours or a partner's), what group or groups they belong to, whether they are logged on, and so on.

For example, if the end user is not logged on, there is just one menu item to initiate an enrollment process (along with the log-on link). If the end user is logged on and is an internal business user, the user sees a set of service request options and a list of their deployed partner connections.

External partner users just see an option to request a new connection. One menu item in the Userview is the inbox, where open tasks are listed for the particular end user. Users click the inbox, then click the open task in the inbox to access their pending activity in the workflow, typically exposing a form to fill out or to inspect and approve.

Design for maintainability

Axway strongly encourages you to design applications for ease of maintenance. The potential of RAD can be fulfilled during initial application development. Perhaps more importantly, the benefits of RAD can continue throughout the application's lifecycle. The majority of investment in an application follows its initial development. Designing and documenting for easy change and reuse provides the greatest return on that investment.

Getting started

2

Go to the Application Studio Help Center for information about getting started. There are topics about setting up users and help for designing, running and monitoring applications you create.

The Help Center is at http://app-studio-help-center.squarespace.com/. You also can open the Help Center with a link at the top of the Application Studio user interface.

See *Application Studio overview* on page 10 for an introduction to the product.

REST Form Store Binder plugin 3

The REST Form Store Binder plugin specifies that form field data, upon submission, are packaged as a JSON object and sent via the configured REST service using an HTTP POST operation. The plugin is available in Application Studio 1.4 and later.

This is one of the Application Studio plugins that implement form binders that provide REST access to external web services. The data are not stored in the Application Studio database.

Configuration fields

Application Studio creates REST calls at runtime using the configuration you specify in form properties when designing the application.

When editing a form's properties, select **REST Form Store Binder** as the Store Binder and click **Next**. You must complete fields on three configuration pages. Click **Next** to advance to the next page.

REST host configuration

Server Name / IP Address

The fully qualified domain name or IP address of the server the plugin is connecting to. You can use **localhost** only if Application Studio and the server are on the same computer.

Port

The port the server listens for connections.

Root Path

The directory path shared by all REST services on the server. For example, `/rest/services`.

User Name

The user name to perform HTTP basic authentication.

User Password

The password to perform HTTP basic authentication.

Disable Secure Connection

Connect without a secure connection over HTTP.

Email Administrator on Error

Specifies whether to send an email message to the Application Studio administrator when an error occurs. See *Error handling* on page 26 for details.

Target URL configuration

Path of REST API Service

The path of the REST service to call. The path is appended to the specified *Root Path* on page 16.

Additional REST Query Parameters

Optionally, specify name-value pairs to be sent each time the form is executed.

For example, you might need to specify additional information to the web service, such as where to store the JSON data with the parameter **jsonfile=filename.json**.

JSON field binding configuration

JSON field bindings

Enter field ID-JSON tag pairs. See *JSON tag syntax* on page 19 for details.

At application execution time, the REST call is constructed by the binder using all of the JSON field bindings information. The Root Path and Path of REST API Service are concatenated. Basic Authentication is used with the specified user name and password.

You must know how the REST service uses the JSON object to design the object properly with JSON tags.

Configuration, form entry, REST call examples

The following are an example configuration of the REST Form Store Binder plugin, an example of data entered by a form user, and an example a REST call constructed and executed by the plugin on form submission.

Configuration

- Server name / IP address: restserver
- Port: 8101
- Root path: /b2bi
- User name: admin
- User password: Abc123@

- Do not disable secure connection
- Email administrator on error: Enabled
- Path of REST API service: /contacts/add
- Additional REST query parameters: param1=1, param2=2
- JSON field bindings (field ID-JSON tag pairs):
  - name: NAME
  - street: $.ADDRESS.STREET
  - city: $.ADDRESS.CITY
  - state: $.ADDRESS.STATE
  - zip: $.ADDRESS.ZIP

Form entries

- Name: Joseph Smith
- Street: 123 Anystreet
- City: Anycity
- State: Anystate
- Zip: 12345

Labels and field IDs in the form are:

- Name: name
- Street: street
- City: city
- State: state
- Zip: zip

REST call

- Final URL: `https://restserver:8101/b2bi/contacts/add?param1=1&param2=2`
- Basic authentication header: `Authorization: Basic YWRtaW46QWJjMTIzQA==`
- JSON in POST Body:

```
{
"NAME": "Joseph Smith",
"ADDRESS": {
"STREET": "123 Anystreet",
"CITY": "Anycity",
```

```
"STATE": "Anystate",
"ZIP": "12345"
}
}
```

JSON tag syntax

A binder JSON tag uses a hybrid JSONPath subset syntax for JSON object construction. Specify a JSONPath using a leading $ and standard JSONPath dot notation or bracket notation for designating child objects up to the ending name. A JSON tag specifies the object/child path names (and therefore nested position) to be used for the field ID value in the final JSON object.

Note Some form field identifiers are reserved for use by Application Studio. You cannot use them. The reserved identifiers are: **appId**, **appVersion**, **userviewId**, **menuId**, **key**, **embed**.

In the JSON field bindings example in *Configuration, form entry, REST call examples* on page 17, the field ID **street** is mapped to the JSON tag **$.ADDRESS.STREET**. This specifies the form field value for street is placed into the final JSON structure in object ADDRESS, as the value for ending child name STREET. The city, state and zip mappings then add to the ADDRESS object similarly. A JSON tag can contain any characters other than double quote " or backslash \. The path must select a single value that cannot be a JSON object or array.

The same JSON tags using equivalent JSONPath bracket notation instead of dot notation appear as in the following table:

Field ID	JSON tag
name	NAME
street	$['ADDRESS']['STREET']
city	$['ADDRESS']['CITY']
state	$['ADDRESS']['STATE']
zip	$['ADDRESS']['ZIP']

Dot and bracket notation can be mixed arbitrarily as in the following table.

Field ID	JSON tag
name	NAME
street	$.ADDRESS.STREET

Field ID	JSON tag
city	$['ADDRESS'].CITY
state	$.ADDRESS['STATE']
zip	$['ADDRESS']['ZIP']

The leading $. or $ in $[are optional. When using bracket notation, the JSON tag can contain any characters other than single quote ', double quote " or backslash \. The name must select a single value that cannot be a JSON object or array. Another mapping example is in the following table.

Field ID	JSON tag
name	NAME
street	ADDRESS.STREET
city	['ADDRESS'].CITY
state	$['ADDRESS']['STATE']
zip	$.ADDRESS.ZIP

The names for the objects and name-value pairs in the JSON structure are arbitrary; they do not have intrinsic relations to the field ID's they are mapped to. In the preceding example, the field IDs were mapped to upper-case versions of the field IDs, but this is not required. A field ID can be mapped to a path incorporating any JSON name. However, the JSON names and JSON structure must be expected and readable by the REST form load binder receiving the REST call. Both fields IDs and JSON names are case sensitive.

A REST Store Binder JSON tag uses a subset of JSONPath, which includes only the following features:

- Optional leading $. or $ in $[
- Component names in dot or bracket notation
- [*] specifier for grid form objects mapped to arrays. See *Mapping grid objects* on page 20.

Mapping grid objects

In Application Studio, grid objects have columns (fields) and rows, where each row represents a data record. The types of grids you can select in the Form Builder user interface are:

- Simple grid in which grid field values are entered in the form directly.
- Form grid in which a second form is used to enter values for each record.
- List grid in which rows are entered by selecting from a datalist.

Because grids differ only in the way data is entered, the grid fields (columns) are specified the same way for any grid type in a JSON tag. The JSONPath subset syntax for specifying storage of a grid of values in the JSON object is the [*] designation on the grid name.

For example, for mapping a column with form field ID **f1** in a grid with ID **g1**, use:

```
g1.f1 $.G1[*].F1
```

Then to specify column with field ID **f2** also in **g1**, use:

```
g1.f2 $.G1[*].F2
```

According to these specifications, if the grid has three rows of data when the form is submitted, the following JSON is constructed.

```
{
"G1": [
{
"F1": "data1",
"F2": "data2"
},
{
"F1": "data3",
"F2": "data4"
},
{
"F1": "data5",
"F2": "data6"
}
]
}
```

In the preceding table, `data1` and `data2` are the values of row 1, and so on. Each row is an object in the ordered JSON array G1. This processing applies to any simple grid, form grid or list grid. The grid types differ only in the way rows are selected into the grid in the form. All types use the identical mapping technique and produce the same JSON as in the preceding example. The notation [*] informs the binder that G1 is to be an array in the final JSON object. This is the only case where array ([]) designation is used in store binder mappings.

Grid properties can specify certain load and store binders, but grids cannot specify the REST Form Store Binder plugin. Any load and store binders specified for a grid are executed first, then the containing form's REST Form Store Binder. If no store binder is selected in a grid's properties, the grid data is provided as a form field to the containing form's REST call, and no grid data is written to the Application Studio database.

The names for the objects and name-value pairs in the JSON structure are arbitrary; they do not have intrinsic relations to the field ID's they are mapped to. In the preceding example, the field IDs were mapped to upper-case versions of the field IDs, but this is not required. A field ID can be mapped to

a path incorporating any JSON name. However, the JSON names and JSON structure must be expected and readable by the REST form load binder receiving the REST call. Both fields IDs and JSON names are case sensitive.

Mapping subforms

You can include standard, multipaged and Ajax subforms within a form in Application Studio. This topic describes including subforms in the REST JSON object. This documentation assumes you know how to include subforms within forms.

Also see *Subform binder execution order* on page 28.

Standard subform

To include fields of a standard subform within a main form into the REST JSON object:

1. Specify **REST Form Store Binder** as the Store Binder in the main form.
2. Specify **Parent Form Binder** as the Store Binder in the subform's container in the main form. The Store Binder specified in the subform's properties itself is overridden by the container specification.
3. In the main form mapping, include the qualified field names of the subform.

For example, if the main form has person contact data and an organization subform with the container name **subform1**, and the field mappings include all data on the main and subform, the following table represents appropriate mapping.

Field ID	JSON tag
name	NAME
street	$.ADDRESS.STREET
city	$.ADDRESS.CITY
state	$.ADDRESS.STATE
zip	$.ADDRESS.ZIP
subform1.org	$.SUBFORM1.ORG
subform1.div	$.SUBFORM1.DIV
subform1.ext	$.SUBFORM1.EXT

Further, suppose the data in the following table were entered when the form was submitted.

Field	Value
Name	Joseph Smith
Street	123 Anystreet
City	Anycity
State	Anystate
Zip	12345
Org	Axway
Div	Marketing
Ext	3001

Then the following would be the JSON sent to the REST call.

```
{
"NAME": "Joseph Smith",
"ADDRESS": {
"STREET": "123 Anystreet",
"CITY": "Anycity",
"STATE": "Anystate",
"ZIP": "12345"
},
"SUBFORM1": {
"ORG": "Axway",
"DIV": "Marketing",
"EXT": "3001"
}
}
```

Note that the JSON tag naming is arbitrary. For instance, you could put all the JSON data at the top level by using the following mappings.

Field ID	JSON tag
name	NAME
street	STREET
city	CITY

Field ID	JSON tag
state	STATE
zip	ZIP
subform1.org	ORG
subform1.div	DIV
subform1.ext	EXT

The **subform1.** qualification still selects the field data out of the subform. The JSON structure must conform to what the web service expects.

Multipaged subform

To include fields of all pages of a multipaged subform within a main form in the REST JSON object:

1. Specify **REST Form Store Binder** as the Store Binder in the main form.
2. Specify **Parent Form Binder** as the Store Binder in the properties of each subform (page). There is no binder specification allowed in the multipaged subform container in the main form.
3. Do not specify the multipaged subform container advanced option **Partially store form when page changed.**
4. In the main form mapping, include the qualified field names of each subform (page).

The following is an example of mappings with a multipaged subform of two pages.

Field ID	JSON tag
name	NAME
street	$.ADDRESS.STREET
city	$.ADDRESS.CITY
state	$.ADDRESS.STATE
zip	$.ADDRESS.ZIP
multipaged1.org	$.MULTIPAGED1.ORG
multipaged1.div	$.MULTIPAGED1.DIV
multipaged1.ext	$.MULTIPAGED1.EXT

Field ID	JSON tag
multipaged2.title	$.MULTIPAGED2.TITLE
multipaged2.grade	$.MULTIPAGED2.GRADE
multipaged2.code	$.MULTIPAGED2.CODE

Further, suppose the data in the following table were entered when the form, including the two subform pages, was submitted.

Field	Value
Name	Joseph Smith
Street	123 Anystreet
City	Anycity
State	Anystate
Zip	12345
1	
Org	Axway
Div	Marketing
Ext	3001
2	
Title	Analyst
Grade	004
Code	227

Then the following would be the JSON sent to the REST call.

```
{
"NAME": "Joseph Smith",
"ADDRESS": {
"STREET": "123 Anystreet",
"CITY": "Anycity",
"STATE": "Anystate",
```

```
"ZIP": "12345"
},
"MULTIPAGED1": {
"ORG": "Axway",
"DIV": "Marketing",
"EXT": "3001"
},
"MULTIPAGED2": {
"TITLE": "Analyst",
"GRADE": "004",
"CODE": "227"
}
}
```

Ajax subform

To include fields of an Ajax subform within a main form in the REST JSON object:

1. Specify **REST Form Store Binder** as the Store Binder in the main form.
2. Specify **Parent Form Binder** as the Store Binder in the subform's container in the main form. The Store Binder specified in the subform's properties itself is overridden by the container specification.
3. In the main form mapping, include the qualified field names of the subform.

The example mappings in *Standard subform* on page 22 are the same for an Ajax subform. The Ajax subform must be configured properly in the main form using a select box control to select values into the subform before form submission.

Error handling

The REST Form Store Binder plugin detects the following general categories of errors:

1. Certain design-time validation errors in the binder configuration fields (in Form Builder Properties). For example, a validation error is displayed if the REST HTTP port is not in the range 1-65535.
2. Run-time configuration failures. For example, a non-existent field ID in the mappings table: Configuration failure. Please contact your administrator.
3. Run-time communication failures. For example, a REST server authentication failure: Communication failure. Please contact your administrator.

The second and third errors are displayed when end-users are completing forms. This is why the messages advise contacting an administrator of the application.

Email messages about errors are sent to the Application Studio administrator when all of the following conditions are met:

- The plugin option **Email Administrator on Error** is enabled.
- The Application Studio administrator account has a valid email address.
- SMTP notifications are configured properly in system settings.

Configuration failures can generate email messages for configuration or system errors or both.

Sample email message 1

Subject: Configuration Error

Body: com.axway.appstudio.core.extension.rest.BinderConfigurationException: Invalid form field Id: TextField

Sample email message 2

Subject: Communication Error

Body: com.axway.appstudio.core.common.rest.RestException: HTTP 401 - Authentication Required

Sample email message 3

Subject: System Error

Body: javax.ws.rs.ProcessingException: org.apache.http.conn.HttpHostConnectException: Connect to restserver:port [restserver/ip-address] failed: Connection refused: connect

Additional information

The following additional information relates to use of the REST Form Store Binder plugin.

Unsupported form controls

The REST Form Store Binder plugin does not support the following form controls:

- Custom HTML
- File upload
- Image upload

Large text data that needs to be uploaded (for example, a certificate) should be copy-pasted into a text area field.

Workflow variables

The REST Form Store Binder plugin does not affect usage of workflow variables. If workflow variables are associated with form fields that are mapped to the REST service, data are written to the workflow variables in addition to being placed into the REST JSON object by the binder.

Handling of integers, booleans, null values in JSON

Field values in forms are stored as string data in Application Studio, including integer and boolean values. When the REST Form Store Binder plugin sends string values for fields, it creates JSON with integer and boolean values in quotes, as in the following example.

```
{
"enabled": "true",
"maxConcurrentConnections": "100",
"retries": "5",
"holdMessagesForPickup": "false",
"port": "21"
}
```

The web service target of the REST call must make the necessary conversions. For example, change string "true" to boolean true, string "100" to integer 100, and so on.

Note that form fields left empty by users have null values, and the plugin does not include the fields in the JSON. The receiving web service can set the missing fields to whatever values the web service uses for null values.

Axway API Server can be setup to accept REST calls before reaching the target web service and convert values appropriately.

Subform binder execution order

In a nested subform structure, subform binders are executed generally from innermost to outermost. Consider the following definitions:

- A subform element is a container with a child form element.
- A multipaged form is a container with multiple child form elements.
- Every form element has its own load and store binders (default to Workflow Form Binder), unless you remove them from the properties when designing the form.
- The load and store binders in a subform element properties overrides the load and store binders of its child form element. Overridden load and store binders are ignored during execution.

- A section element can have its own load and store binders. If **Workflow Form Binder** is selected for a section element, the table of a main form is used.
- Every load and store binder in a form is executed.

For example, if the subform and store binders structure of a main form is:

Main form

- Subform 1
- Subform 2
  - Inner subform 1
  - Inner subform 2
- Section 1
  - Subform 3
    - Multipaged child form 1
    - Multipaged child form 2
    - Multipaged child form 3 (store binder is removed)
- Section 2

Then the store binder execution sequence is:

1. Subform 1
2. Inner subform 1
3. Inner subform 2
4. Subform 2
5. Multipaged child form 1
6. Multipaged child form 2
7. Subform 3
8. Section 1
9. Section 2
10. Main form

Although the REST Form Store Binder plugin displays in the Store Binder selection list for a form section, it is not supported for a section. Selecting it for a section causes a configuration error.

Batch Process Tool plugin 4

Application Studio has a plugin that enables you to execute workflow processes against any number of datasets. The plugin is the Batch Process Tool. After logging on to the administration user interface, you can find it in the list of plugins at **System Settings > Manage Plugins**.

A dataset is a row of data in a database table. For example, each row might represent contact information for a would-be business partner.

You can use the Batch Process Tool to drive a workflow process for enrolling, or onboarding, partners. Although this is the primary intended use, you can use the plugin with any workflow process where flexibility is required for the number of datasets.

Used in a partner enrollment application built in Application Studio, the plugin can manage launching the workflow process using data of would-be partners, including their email addresses. For example, the process might send emails containing URL links to would-be partners. The email recipients could click the links in the messages to access enrollment forms to complete and submit.

Users of Application Studio can build enrollment applications meeting their specifications, with or without help of Axway professional services consultants. For example, a user might want to build an enrollment application for onboarding partners in a gateway product such as Axway SecureTransport.

There are two phases in an onboarding process using Application Studio:

1. **Design and development**. Application developers design and build the enrollment application in Application Studio. The workflow includes the Batch Process Tool.
2. **Execution**. Relationship managers run the published application to collect enrollment data from targets.

Design and development guidelines

To use the Batch Process Tool, you must build a workflow in Workflow Designer that has a driver process and a target process. You then use the administration user interface to add a target datalist, a batch driver form and a userview.

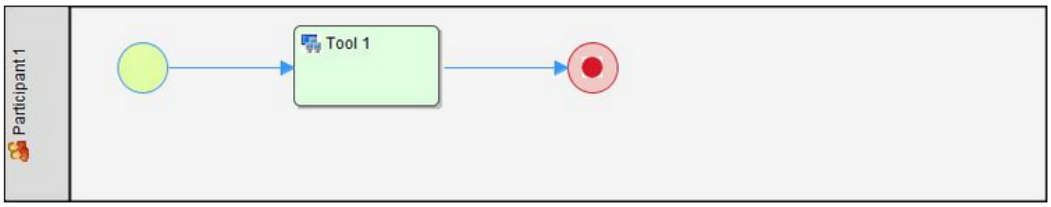
The example in this section illustrates the essentials of using the Batch Process Tool. You can later enhance the example into a complete enrollment application.

Driver process

When executed the driver process must display a form (the batch driver form) with a list of records to choose from (the list grid). There also must be a start batch button to launch the batch of target processes on the records. The driver process must have a tool element, to which the Batch Process

Tool is mapped and configured.

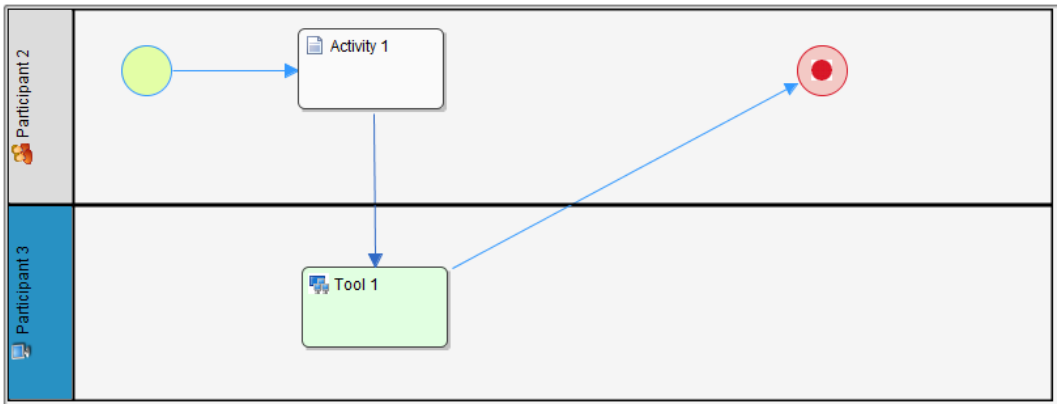
The following graphic illustrates an example of the driver process in Workflow Designer.



Target process

The target process can be any valid process. The Batch Process Tool, running in the driver process, launches an instance of the target process for each record in the list grid when the user clicks a start batch button. The target process defines a set of workflow variables. When the Batch Process Tool launches the process instance, values are passed from the record's columns to the workflow variables to be used by the target process (see *Target datalist* on page 31).

The following graphic illustrates an example of the target process in Workflow Designer.



When the workflow is completed and deployed, an application developer assigns permissions to each process, specifying who can run it. For example, the permissions could be set to Role = Logged in User.

Target datalist

You need a datalist with an **id** column and columns of data representing values passed to launched processes. The datalist is named **TargetDatalist** in this documentation. When the driver form is displayed, the user selects some or all of the data records in the TargetDatalist into the list grid. The Batch Process Tool then launches processes for records in the list grid, one row per process instance.

For example, the datalist could have the following columns:

- id
- firstName
- lastName
- company
- email

You map these columns to workflow variables in the target process when configuring the Batch Process Tool. Values of the columns in one datalist row pass to the corresponding workflow variables of that process instance. The datalist's source of data could be any form table in the Application Studio database or any database specified via a JDBC connector. Data also can be imported from a CSV file with an import control described in *Userview* on page 35.

The Batch Process Tool requires the **id** column for visibility purposes; it is not passed to the target process instances.

Upon clicking a start batch button, the processes are launched using the selected datalist records. After launch, the selected records are stored in a post-launch database table.

Batch driver form

A batch driver form is required. The following are examples of the property values for the form.

- ID = driverForm
- Name = DriverForm
- Database table name = ds_driverform

A batch driver form needs the following elements.

Batch name text field

Specify the Default Validator to make the field mandatory. The field value uniquely identifies this batch of processes.

List grid

For the list grid select the relevant columns from the TargetDatalist. You do not have to specify all of the datalist columns in the list grid. You only need the relevant values or values you intend to map to the target process. For example, you can use firstName, lastName and email.

Filters specified in the TargetDatalist display in the list grid add entry dialog, which selects rows from the datalist.

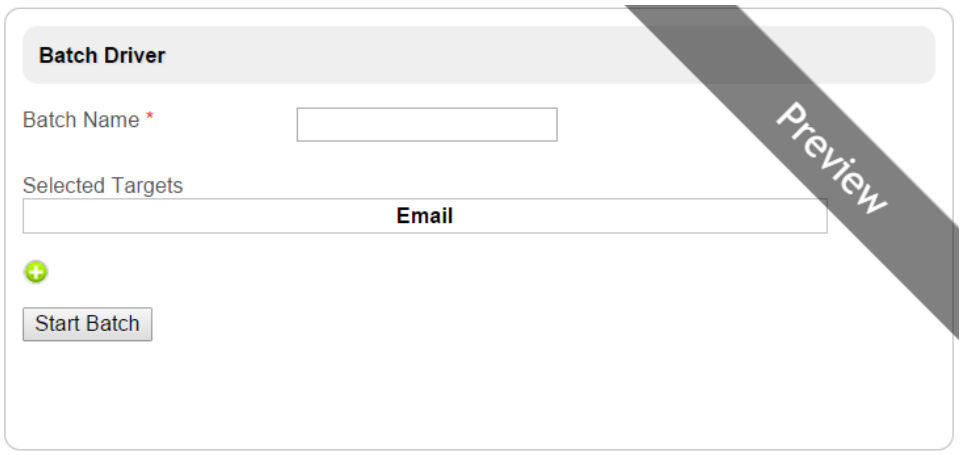
Custom HTML field

Use the JavaScript validator in a custom HTML field for a Start Batch button. The validator also performs client-side checking when a form user clicks the button. For example, it checks whether the batch name is unique. It also displays messages when it detects errors.

You can download an HTML file containing the Start Batch button and validator script. You can view the page source to copy the script.

http://app-studio-help-center.squarespace.com/storage/batch_plugin/custom-html-js-validator.html

The following graphic illustrates an example of the batch driver form.



Selected targets form

The selected targets form is for storing records that were executed in the batch. The following are examples of the property values for the form.

- ID = formSelectedTargets
- Name = FormSelectedTargets
- Database table name = ds_formSelectedTargets

The form must have a process ID text field, with ID = processId, and one text field for every column selected to be displayed in the batch driver form's list grid, using the same field IDs..

You must specify the selected targets form in the selected targets list grid of the batch driver form with the following properties:

- Store binder = Multirow Form Binder
- Configure Multirow Form Binder form = FormSelectedTargets
- Foreign key = processId

CSV form

The CSV form is used for importing data of candidate partners from CSV files. The data can be imported with a CSV Import control as described in *Userview* on page 35.

The following are examples of the property values for a CSV form.

- ID = formCsv
- Name = FormCsv
- Database table name = ds_formcsv

Your form must have a text field for each record column, or row column, in the source CSV file.

The CSV Import control in your userview must:

- Specify your CSV form
- Map field IDs to column numbers, starting with column number 0
- Set the Start From Row Number field to 1

In this CSV example, the following is the path the source data takes:

File on the file system >

Table ds_formcsv (CSV Import) >

Target datalist >

Batch driver form List grid >

Table ds_formselectedtargets (after batch launch)

Use a consistent set of field IDs. These are:

- Specified when creating the CSV form
- Available for the canvas in designing the target datalist
- Specified in the column mapping in the batch driver form list grid configuration. Note that not all form fields must be selected in the list grid.
- Specified when creating the selected targets form.

Using a CSV form and CSV Import control are one way to make data records available to the application for batch execution. Alternatively, you can use a CRUD control in the userview to create the data records manually. This would require using a CRUD control in the userview and a different type of form.

You also could configure the target datalist to load data directly from any Application Studio table or any database specified via a JDBC connector (see *Target datalist* on page 31).

Mapping

Map the batch driver form to the driver process as a whole. That is, map to the default Run Process activity.

Map the Batch Process Tool to the tool element in the driver process. When it is mapped, the following required values must be configured. You select items 1-4 from drop-down lists.

1. Batch driver form name
2. List grid ID
3. Batch name field ID

4. Target process name
5. Workflow variable mapping specifies the list grid (TargetDatalist) column name mapping to workflow variable mapping. You can specify up to six mappings.

Userview

Add a userview with:

- HTML home page.
- Run process control set to run the driver process. It displays the batch driver form when clicked.
- Inbox assigned to the target process. This is for workflow activities after a process has been launched.
- Optional CSV file import control to upload records to the TargetDatalist. The first row of the CSV file should be a comma-separated list of column names for the data.
- Optional list control assigned to view the TargetDatalist's form table. This is a direct view of data imported into the database.

The following graphic illustrates an example of the userview.

Optional visibility features

The previous topics describe building essential elements for the Batch Process Tool. However, you can add visibility features to the userview for tracking the status of batches you have started.

You need the following additional datalists to enable visibility:

- A batch process status datalist to view records in the batch table. It shows a list of all batches that have been started or completed. For example, the datalist can be named

BatchProcessStatus.

- A detail list datalist to view the process records of a specific batch. Each datalist row has a link to launch a detailed view of the batch's processes. For example, the datalist can be named **DetailList**.
- A show-all datalist to view process records of all batches in one aggregate view. For example, the datalist can be named **ShowAllList**.

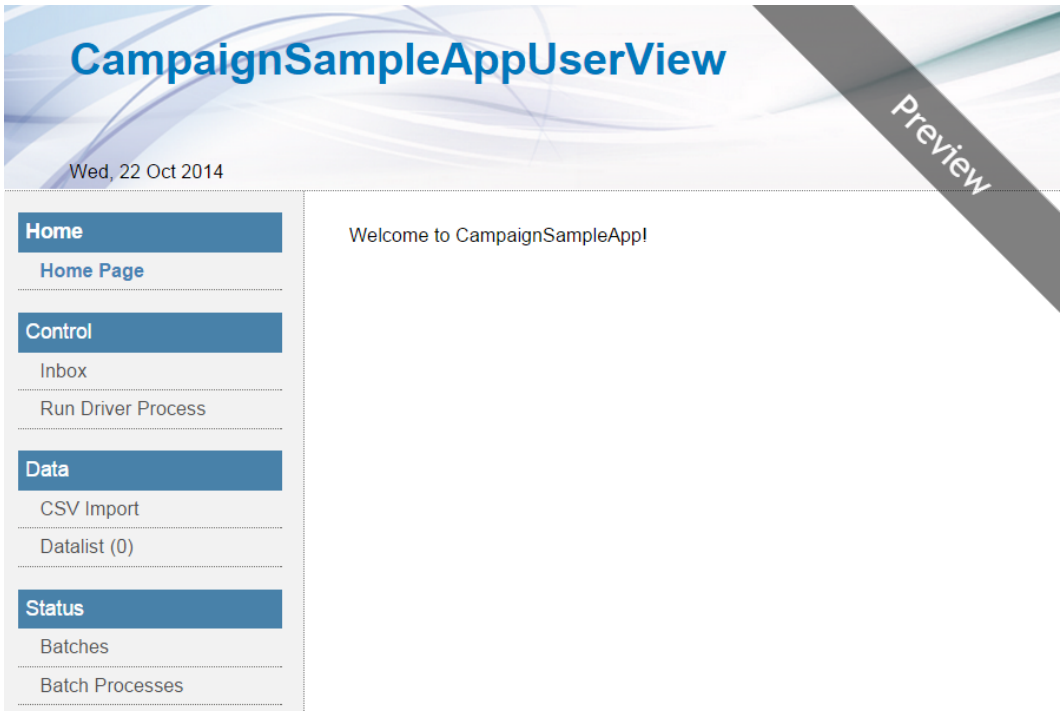
The DetailList and ShowAllList datalists must execute custom SQL queries to retrieve data for viewing. You can download a ZIP file containing both SQL queries.

http://app-studio-help-center.squarespace.com/storage/batch_plugin/BatchProcessToolSQLqueries.zip

With these changes the userview adds the following categories:

- A status category with controls to view the BatchProcessStatus and ShowAllList datalists.
- A hidden category with a list control for the DetailList datalist. A hidden property is applied to this category. This is because it must be in the userview to view a datalist, but the user does not access the DetailList directly, but only by taking the link from the BatchProcessStatus batch record.

The following graphic illustrates an example of the userview with the optional visibility features.



Publish application

Once complete, an application developer must publish the application. This makes it executable.

Running the application

It is presumed a relationship manager runs the published application that uses the Batch Process Tool. The relationship manager runs the application by:

1. Getting the URL for the application's userview from the application developer.
2. Opening the application's userview URL and logging on.
3. Using the CSV import control to upload data records into the TargetDatalist datalist or populating it another way.
4. Clicking the run driver process control to launch the batch driver process.
5. Assigning a unique batch name for the set of instances to be launched.
6. Selecting all or subset of records from the TargetDatalist datalist into the list grid of records.
7. Launching the batch by clicking the Start Batch button.

When the batch driver process is launched, a form is displayed with no value in the batch name field and an empty list grid is displayed, as illustrated in the following graphic.

The relationship manager specifies a batch name and loads the list grid, optionally using any filter from the associated datalist, and clicks **Start Batch**. When submitted, a dialog is displayed. On it is the name of the target process and how many instances will be launched. The relationship manager clicks **OK** to proceed.

The Batch Process Tool launches the target processes, one process instance for each row in the list grid. Values are copied from the list grid row to the process workflow variables, according to the mapping configured in the plugin.

No restrictions are placed on the target processes. They can perform any operations that any Application Studio process can perform. What they do is beyond the control of, and not relevant to, the Batch Process Tool.

The batch of launched target processes execute within the current userview used to run the batch driver process. Once a target process is triggered, the continuous activity goes to the user's inbox in the userview, based on the participant mapping provided, and the user can view the activity using any of the configured inbox options.

Note also that beyond the columns and filters in the TargetDatalist datalist, the source of the data in the columns, datalist configuration parameters and extended list grid configuration parameters are

irrelevant to the Batch Process Tool. What happens to the datalist itself after the batch is launched is not relevant to the Batch Process Tool. The datalist can have rows deleted or itself be deleted, it does not matter to the launched batch.

When used to perform onboarding, each target process instance would send an email to a would-be partner. The email would contain a link to a form in the target process, which the email recipient would complete and submit. The workflow would then continue, posting the submission to the relationship manager's userview inbox, where the relationship manager would approve or reject the submission. On approval, a final workflow step would be designed to create a user account in Application Studio for the new partner.

Additional information

1. You can assign a form directly to a process, rather than to an activity within the process. The process starts only upon form submittal. If assigned to an activity, the process starts first and then the form displays when the process flow reaches the form's activity.
2. Workflow variables support only the text string data type. This means there are no issues for type mismatch between column values and variables.
3. Each column value of the selected TargetDatalist is copied to a workflow variable according to the configured mapping. Column names must be unique in the TargetDatalist. Workflow variable names must be unique in the process and are case sensitive. In addition, column names might be case sensitive, depending on the data source.
4. Any filters specified in the TargetDatalist are available in the list grid add entry dialog that selects TargetDatalist rows to load the list grid. The list grid is used to select rows from the TargetDatalist on which to launch the processes.
5. The target process can define workflow variables not mapped to TargetDatalist columns. The process executes with mapped values copied. The process uses non-mapped workflow variables for other purposes.
6. The list grid is configured to record the selected records in a post-launch database table, recording the values used to launch each specific process instance. Status of each process instance, recorded in a database table, is available to reports and dashboards. The database table is named **aas_bpt_process**.
7. The Batch Process Tool performs error checks on browser clients. The JavaScript validator in the custom HTML element in the batch driver form enables the error checks. It generates messages in pop-ups when errors occur.
8. The Batch Process Tool performs error checks on the Application Studio server as a backup to the error checking performed by the JavaScript validator. Server-side-detected errors are written to log files and emailed to the user who is running the application. The Batch Process Tool checks whether:
    - Datalist column names are mapped to workflow variables.
    - Mapped workflow variable names are correct.
    - Two or more TargetDatalist columns are not mapped to the same workflow variable.

- A unique batch name is specified.
- At least one data row is selected for the list grid.
- The number of list grid rows does not exceed 150.

9. The list grid can ensure unique rows are selected into it by specifying its unique column property.
10. The list grid inherits any filters specified in the source TargetDatalist, making the list grid searchable.
11. The list grid has a sort property. This allows manipulation by row, but not sorting all rows in one operation.
12. If an encrypted field is selected into the list grid, the encrypted value forwards to the workflow. There is no decryption. The application designer must ensure the TargetDatalist does not contain sensitive data.

Email plugin with status options

5

Application Studio has an email plugin that enables you to specify variables that trigger actions in workflows based on the success or failure of email messages sent to users of your applications. The tool is named Axway Statusing Email Tool. This plugin also offers enhanced security over the other email tool available to use in worflows, the Email Tool plugin. However, the key advantage of the Axway Statusing Email Tool is its variables mapping option.

The tool also enables you to specify a workflow variable to which the reason for failure could be mapped. If the `wf_email_reason` is specified in the status, this variable holds the exception message when there is a failure. This would help in troubleshooting.

The following are the configuration fields when you select the Axway Statusing Email Tool for use in a workflow.

Configure Email Tool page

SMTP Host

Server host name.

SMTP Port

Server port number. If not TLS, the port typically is 25 for an SMTP server. If TLS, this must be the port to connect to the server via TLS.

Note If you want to connect to an external server via TLS, and the external server uses a self-signed certificate or a certificate issued by your internal CA, see *Certificate management tools* on page 58 for implementation details.

Security

Optionally, select TLS the connection security protocol.

SMTP Username

If required, the user name for connecting to the server.

SMTP Password

If required, the password for connecting to the server.

Email page

From

Sender address for all outgoing email notifications.

To

Email addresses of the recipients, separated by semicolons. For example: person1@xyz.com; person2@xyz.com.

CC

All copied email addresses for outgoing messages, separated by semicolons. For example: person1@xyz.com; person2@xyz.com.

Subject

Subject of the message.

Message

Body of the message.

Attachments page

Adding attachments to emails is optional.

Form

Name of the form to attach to the email.

Form Upload Fields

File attachment field ID. You must specify a form for this option to work.

Files

If files are not in the form, you can specify the path to the files you want to attach.

Workflow Variables Mapping page

You can use the Status and Reason fields independently of each other.

Status

Variable to set the status of the sent email messages.

Reason

Variable to set additional information about the email messages.

Database case-sensitivity rules

6

You must follow guidelines to make sure the applications you develop in Application Studio are database agnostic. This ensures portability of applications among instances of Application Studio regardless of the database type. For example, an application developed in an instance of Application Studio with an Oracle database can be exported and imported to an instance of Application Studio with a MySQL database, and vice versa.

Regardless of the database type, apply the following rules when creating a database, tables and column names and in custom queries:

- For database names and table names, mixed case is acceptable, but the names must be case sensitive.
- For database column names, use all upper-case.

The following are the case requirements for database objects by database type and operating system:

For database names and table names

MySQL - case sensitive

Oracle - the Application Studio hibernate layer allows case insensitive

For database column names

MySQL - case insensitive

Oracle - upper-case

Form features 7

Application Studio enables designing forms that users of your applications can complete and submit to provide you with business and technical data crucial for fostering business-to-business transactions. The following topics describe features to know about when designing forms.

File upload limit for forms

You can build forms that enable users of your application portals to attach files. Efficient use of the attachment feature requires knowing about its limitations for maximum file sizes.

Two file-upload limits govern attaching files to forms.

Per-form limit

Application Studio allows users to attach files totaling a maximum of 50 megabytes per form. This maximum includes not only attachments, but also the data a user enters on a form. The per-form maximum is a hard limit and is not configurable.

Per-attachment limit

Application Studio allows you specify the maximum size of each file attached to a form. The default per-attachment maximum is 50 MB, which is the same as the per-form limit.

If you are designing forms that ask end-users to attach multiple files per form, best practice is set the per-attachment limit lower than 50 MB. This reduces the risk that users might try to attach multiple files that together exceed the 50 MB per-form limit.

For example, if you have a form that asks users to upload five file attachments, you can set the per-attachment limit to 10 MB. If a user tries to upload a file larger than 10 MB, Application Studio displays an error message and prevents the user from submitting the form.

The per-attachment limit can be set in the administrator user interface. Select **System Settings > General Settings** and scroll down to the File Upload Size Limit (MB) field. Enter a value less than 50 MB and click **Submit**.

Save as draft option

Forms you design for your applications by default have buttons that let application users save partially completed forms so they can return later and complete and submit the forms. You can disable the save-as-draft button on selected forms.

1. Select the application you want to edit on the Design Apps menu in the administration user interface.

2. Select a process and then the **Map Activities to Forms** tab.
3. Select **Remove Save as Draft Button** to remove the button from the form you want.

Encrypt text field data

Text fields can be encrypted in the database, but still display as plain text to users of forms. Encryption is enabled by text field. A form can have a mix of encrypted and plain text fields.

1. Select the application you want to edit on the Design Apps menu in the administration user interface.
2. Select **Forms & UI** to display the list of forms in the application.
3. Click a form name to edit the form.
4. Place the cursor over the text field to encrypt and click **Edit**.
5. Select the **Encryption** check box and click **OK**.

Password policy and global SMTP settings

8

The Security Enhanced Directory Manager plugin enables you to:

- Configure password policies
- Configure a global SMTP server for email notifications

Another function, setting up an LDAP directory, is described in *LDAP user authentication* on page 50.

Application Studio can send email notifications to users for account creation, password reset and other events. Doing this requires configuring a connection to an SMTP server and setting up email templates.

For security purposes, all passwords are hashed and stored encrypted.

Configure password policy

To configure the password policy, select **System Settings > Directory Manager Settings** and click **Configure Plugin** to open the General page of the plugin configuration.

General page

The following are the fields on the General page.

Show Login info (e.g. Last Login Date)

When checked, a banner is displayed, showing the user's log-on date and time and failed log-on attempts since the user last logged on.

Failed Login Attempts for Account Lockout

Enables locking users' accounts after a specified number of failed log-on attempts. Options are blank and 3 to 10 attempts. Blank means users are not locked out regardless of the number of failed log-on attempts.

Account Lockout Period (Minutes)

Duration of the lock-out interval in minutes. Options are blank and 10 to 60 minutes. Blank means that lockouts are disabled.

Allow Session Timeout (Inactivity Timeout)

Expires a user session after a specified period of inactivity. The default inactivity period is 30 minutes.

This field only lets you enable or disable session timeouts. You must edit a value in the Tomcat`conf/web.xml` file to change the inactivity period. The following is the parameter in the file to change:

```
<session-config>

        <session-timeout>30</session-timeout>

</session-config>
```

If you make any changes to the `web.xml` file, restart Application Studio for the changes to become effective.

Hard Session Timeout (Hours)

Expires a user session after the specified hours of continuous connection, regardless whether a session is active. Available values range from blank to 48 hours. Selecting blank disables hard timeouts.

Click **Submit** to save changes or **Next** to configure password rules.

Saved changes take effect the next time you log on.

Default Directory Password Policy page

The following are the fields on the Default Directory Password Policy page.

Requires Password Change on First Login

Forces a user to change password when logging on the first time.

Generate Random Password

Instructs the system to generate a random password whenever a user account is created, and subsequently to send the generated password to the email address configured for the user

Enable Forgot Password

Enables the Forgot Password link on the log-on page of Application Studio so users can request resetting their passwords.

Forgot Password Link Validity Period (Minutes)

Specifies how long the link to reset a user password is valid. Beyond this period, users must re-request a password reset. The options are blank, 15, 20, 25, and 30 minutes. Blank means that the link remains valid and does not expire.

Number of Unique Passwords Before Re-use

Defines the password re-use policy and specifies how many last passwords cannot be re-used when they change their password. The options are 0 through 10 unique passwords.

Password Minimum Length

Specifies the minimum number of characters a password should contain.

Password Mandatory Characters

Specifies password requirements, such as whether passwords must have at least one upper-case character, one lower-case character, and so on.

Password validity Period (Months)

Specifies how long a password is valid before it needs to be changed. The options are blank, 3, 6, 9, or 12 months. Blank means all user passwords never expire.

Number of days to show the notification before password expiry

Users can be notified in the banner after logging on that their password is about to expire. This option specifies how many days before the actual expiration date the user is notified. The options are blank and 5 to 30 days. If set to blank, users receive no warning and their passwords expire on the prescribed schedules.

Click **Submit** to save changes or **Next** to configure an SMTP server.

Saved changes apply to all users as events occur. For example, if a user's password was last changed five months ago and you change the password validity from six to four months, the change makes the user's password invalid immediately. The new setting is applied to that user the next time the user logs on.

Configure SMTP server and email templates

To configure an SMTP server and email templates, select **System Settings > Directory Manager Settings** and click **Configure Plugin** to open the General page of the plugin configuration. Click Next until the Notification page is displayed.

SMTP settings

The following describes the SMTP fields on the Notification page.

SMTP Host

Server host name.

SMTP Port

Server port number. If not TLS, the port typically is 25 for an SMTP server. If TLS, this must be the port to connect to the server via TLS.

Note If you want to connect to an external server via TLS, and the external server uses a self-signed certificate or a certificate issued by your internal CA, see *Certificate management tools* on page 58 for implementation details.

Security

Optionally, select TLS the connection security protocol.

SMTP Username

If required, the user name for connecting to the server.

SMTP Password

If required, the password for connecting to the server.

From

Sender address for all outgoing email notifications.

CC

All copied email addresses for outgoing messages, separated by semicolons. For example: person1@xyz.com; person2@xyz.com.

HTML Content?

Check this to send email messages in HTML. Enable this only when the email templates contain correctly formatted HTML content. When enabled, Application Studio sends messages as text/html MIME type, and email clients try to render the messages as HTML.

Email template settings

The fields for configuring email templates are below the HTML Content? field on the Notification page. There are templates for:

- User creation - Messages to new users that contain log-on credentials.
- Password reset - Messages containing a new password when a user's password changes.
- Forgot password - Messages containing a new temporary password after a user forgets a password and needs a new one.
- Account lockout - Messages telling users their accounts have been locked and informing when the lockout expires.

You can use the messages with the default configurations or change them as needed. The hash variables in the templates are resolved when the messages are sent.

If users are not receiving emails or there are other problems with messages, search the logs for SMTP to find events or errors related to emails and help in troubleshooting.

User-level password policy settings

9

You can set some password policy options at the user level for granular enforcement. Select **Setup Users > Setup Users**. Click **Create New User** to add a user or click a user and click **Edit User** to make changes.

For new users you can:

- Generate a random password for the user.
- Make the password never expire.

For existing users you can:

- Force the user to change their password the next time they log on.
- Reset a user's password.
- Make the password never expire.

Also see:

- *Configure password policy* on page 45.
- The setup users topics in the Joget KB at http://dev.joget.org/community/display/KBv4/Setup+Users

LDAP user authentication 10

Application Studio provides the following types of user authentication:

- **LDAP Only:** The mode of Directory Manager supported by LDAP. The related configuration allows the administrator to configure how the LDAP information is mapped to internal constructs (for example, by group, by department, by role, and so on). When using this mode, all user management pages in the user interface become read-only because LDAP is read-only to Application Studio.
- **DB Only:** If you use the Secure Directory Manager (the default for Application Studio) and do not configure an LDAP directory, you are effectively running in database-only mode. In DB-Only mode, Application Studio reads the user information from the database. Because the information is in the same database that Application Studio is running against, it has full create, read, update, delete (CRUD) access. You can create users, delete them, edit them, and so on. The user management pages in the UI allow you to create, edit, and delete entries.
- **Combined:** Combined mode is also supported by Secure Directory Manager. Combined mode allows you to merge LDAP and DB authentication; that is, users can be in either LDAP or DB. In this mode, the UI allows you to edit only users who are in the DB. Users in LDAP are read-only. Further, if LDAP is configured in addition to the local DB directory, the DB entities cannot contain elements from LDAP. For example, a local Group cannot contain LDAP users.

User management objects

The following describes objects in the user management structure.

Organization

The top-level grouping representing the entire organization.

Department

Represents an organizational department. It can be hierarchical (for example, a department might have sub-departments). Departments can contain a user designated as head of department (HOD). This feature can be used in the workflow to assign an approver based on department. Departments can only contain users from the parent organization or from no organization.

Grade

Represents the user's job grade (for example, software engineer). Grade is organized as a horizontal grouping of users who all do the same job. Grades can only contain users from the parent organization or from no organization.

Groups

A grouping of users based on common goals, interests or projects.

Set up LDAP

To configure the Directory Manager plugin, select **System Settings -> Directory Manager Settings** to open the Directory Manager Implementation page.

By default, Application Studio runs in DB Only mode, using the configuration present in the Security Enhanced Directory Manager.

The LDAP Directory Manager Plugin does not support the enhanced security features available when selecting **Configure Plugin** in the default Security Enhanced Directory Manager.

LDAP is configured manually, defining what LDAP attribute maps to which internal concept. The plugin configuration consists of successive dialog pages, each pertaining to specific user management entities.

There are two ways LDAP Directory Manager can be configured:

- LDAP in addition to the local (DB) directory management (Combined mode)
- LDAP only

Note If you want to connect to an external server via TLS, and the external server uses a self-signed certificate or a certificate issued by your internal CA, see *Certificate management tools* on page 58 for implementation details.

Configure LDAP in combined mode

1. Select **System Settings > Directory Manager Settings**. The Directory Manager Implementation page is displayed.
2. In the Current Plugin Name section, click **Configure Plugin** for the Security Enhanced Directory Manager. The General page is displayed.
3. Click **Next** three times to open the External Directory Manager page.
4. From the External drop-down menu, select **LDAP Directory Manager** and click **Next**. The Configure LDAP Director Manager page is displayed.
5. See *LDAP fields* on page 52.

Configure LDAP Directory Manager for LDAP only

1. Select **System Settings > Directory Manager Settings**. The Directory Manager Implementation page is displayed.

2. In the Select Plugin section, select the desired LDAP Directory Manager from the drop-down menu and click **Select**. The Configure LDAP Directory Manager is displayed.
3. See *LDAP fields* on page 52.

LDAP fields

The following are the pages and fields for configuring LDAP.

Configure LDAP Directory Manager page

URL

URL of the LDAP server. For example,

```
ldap://<host>:<port>
```

Or, if SSL:

```
ldaps://<host>:<port>
```

Note If you want to connect to an external server via TLS, and the external server uses a self-signed certificate or a certificate issued by your internal CA, see *Certificate management tools* on page 58 for implementation details.

Admin Username (Principal)

LDAP administrator user name. Application Studio uses this user name for binding LDAP to do queries. The user name format is dependent on the LDAP directory. For example, for Active Directory it could be the `sAMAccountName Axway\Administrator`.

Admin Password (Credential)

LDAP administrator's password.

RootDN

Root domain name used when binding as administrator.

Click **Next** to display the User page.

User page

Use the User page to configure how entries in LDAP map to users in Application Studio.

User Base DN

Domain name under which the search filter is applied to find records that identify users.

User Import Search Filter

Search filter to select user records. For example, **(objectClass=person)**.

Attribute Mapping - Username

Identifies the user. For example, **sAMAccountName, userPrincipalName, uid**.

Attribute Mapping - First Name

Identifies the users first name. For example, **givenName**

Attribute Mapping - Last Name

Identifies users last name. For example, **sn**.

Attribute Mapping - Email

Identifies the users email. For example, **userPrincipalName**.

Attribute Mapping - Status

Identifies whether the user is active or inactive.

Attribute Mapping - Time Zone

Users time zone (offset from UTC) as a number from -12 to 12.

Click **Next** to display the Employment configuration page.

Employment page

Use the Employment page to enter additional information about users.

Attribute Mapping - Employee Code

Allows mapping an attribute that represents an employee code.

Attribute Mapping - Job Title

Allows mapping an attribute that represents a job title.

Attribute Mapping - Report To

If the user contains an entry for someone to whom they report, this attribute allows it to be mapped.

Map To "Report To" Entry Attribute

This attribute maps to another user record whose value is Attribute Mapping - Report To. For example, Attribute Mapping - Report To is manager and Map To "Report To" Entry Attribute is **distinguishedName**. In this way, user A has a manager with attribute xyz and reports to the user whose distinguishedName equals xyz.

Attribute Mapping - Groups

If the user contains an entry for a group to which they belong, this attribute allows it to be mapped.

Map to LDAP Group Entry Primary Attribute

If the user contains an LDAP group entry for a group to which they belong, this attribute allows it to be mapped.

Attribute Mapping - Departments

If the user contains an entry for a department to which they belong, this attribute allows it to be mapped.

Map To LDAP Department Entry Primary Attribute

If the user contains an LDAP department entry for a department to which they belong, this attribute allows it to be mapped.

Attribute Mapping - Grade

If the user contains an entry for a grade level to which they belong, this attribute allows it to be mapped.

Map To LDAP Grade Entry Primary Attribute

If the user contains an entry for an LDAP grade level to which they belong, this attribute allows it to be mapped.

Click **Next** to display the Group page.

Group page

Use the Group page to configure how LDAP records map to groups in Application Studio.

Group Base DN

Domain name under which the search filter is applied to find records that identify groups.

Group Import Search Filter

Search filter to select group records. For example, **(objectClass=group)**.

Attribute Mapping - ID

Attribute used as the group ID. For example, **cn**.

Attribute Mapping - Name

Attribute used as the group name. For example, **cn**.

Attribute Mapping - Description

Attribute used as the group description.

Attribute Mapping - Users

Attribute that contains users listed in a given group record. For example, **member**.

Map To LDAP User Entry Primary Attribute

Used with Attribute Mapping - Users, this is the attribute whose value is expected to match the value in **Attribute Mapping - Users**. In this way, user A has a manager with attribute xyz and reports to the user whose distinguishedName equals xyz. For example, **distinguishedName**.

Click **Next** to display the Department page.

Department page

Use the Department dialog to configure how LDAP records map to departments in Application Studio.

Department Base DN

Domain name under which the search filter is applied to find records that identify departments.

Department Import Search Filter

Search filter to select department records. For example, **(objectClass=organizationalUnit)**.

Attribute Mapping - ID

Attribute used as the department ID. For example, **cn**.

Attribute Mapping - Name

Attribute used as the department name. For example, **cn**.

Attribute Mapping - Description

Attribute used as the department description.

Attribute Mapping - HOD

If the department record keeps the head of department (HOD), this attribute identifies the HOD. For example, **manager**.

Attribute Mapping - Users

If the department record keeps a list of the users in the department, this is the attribute that contains those users. For example, **member**.

Map To LDAP User Entry Primary Attribute

Used with Attribute Mapping - Users and Attribute Mapping - HOD, this is the attribute whose value is expected to match the value in **Attribute Mapping - Users** or in **Attribute Mapping - HOD**. For example, **distinguishedName**.

Click **Next** to display the Grade page.

Grade page

Use the Grade page to configure how LDAP records map to grades in Application Studio.

Grade Base DN

Domain name under which the search filter is applied to find records that identify grades.

Grade Import Search Filter

Search filter to select grade records. For example, **(objectClass=grade)**.

Attribute Mapping - ID

Attribute used as the grade ID. For example: **cn**.

Attribute Mapping - Name

Attribute used as the grade name. For example, **cn**.

Attribute Mapping - Description

Attribute used as the grade description.

Attribute Mapping - Users

If the grade record keeps a list of the users in a specific grade, this is the attribute that contains those users. For example, **member**.

Map To LDAP User Entry Primary Attribute

Used with Attribute Mapping - Users, this is the attribute whose value is expected to match the value in Attribute Mapping - Users. In this way, user A has a manager with attribute xyz and reports to the user whose distinguishedName equals xyz. For example: **distinguishedName**.

Click **Next** to display the Admin Role page.

Admin Role page

Use the Admin Role page to identify LDAP users assigned to the Application Studio administrator role. This is most often a group record, but could be any individual user or other record.

Admin Role Base DN

Domain name under which the search filter is applied to find records that identify administrator users.

Admin Role Import Search Filter

Search filter to select admin users records. For example, `(objectClass=administrators)`.

Attribute Mapping - Users

Identifies the admin users. If this is a group record it could be member, or if an individual record it could be **distinguishedName**.

Map To LDAP User Entry Primary Attribute

Used with Attribute Mapping - Users, this is the attribute whose value is expected to match the value in Attribute Mapping - Users. In this way, user A has a manager with attribute xyz and reports to the user whose distinguishedName equals xyz. For example, **distinguishedName**.

Click **Next** to display the Advanced page.

Advanced page

The Advanced page has two settings to aid with debugging. If LDAP is not working as expected, enable debug mode and review the log files to help in troubleshooting.

Click **Submit** to save your changes.

Administrator UI access when LDAP is down

If the LDAP server is unavailable or configured incorrectly in Application Studio, you cannot log on to the administrator user interface with a user on LDAP, as the list of LDAP users is not available. The default Application Studio admin user also cannot log on. However, there is a safeguard that enables you to log on if LDAP is unusable.

As part of the LDAP configuration in Application Studio, you must enter the user name and password for connecting to the LDAP server. You can log on to Application Studio with these credentials in the event of LDAP failure. Once logged on, you can correct the configuration issue if that was the cause of the failure.

Certificate management tools 11

Application Studio has script files in the `<install directory>\cli-tools` directory for performing certificate imports and maintenance. All events related to use of the tools are written to `<install directory>/logs/appStudioCli.log`. The scripts are interactive and execute without parameters.

The following topics describe the tools.

Public certificate management

You can use the public certificate management tool to import and manage public key certificates in the Application Studio truststore. Supported file formats are PEM encoded DER (.pem) and binary DER (.cer, .crt, .der) X.509 certificates. The script, named `manageTrustedCerts.sh`, can perform the following functions.

Import

The script can import public key certificate files.

- The script prompts for the certificate file to import and an alias to uniquely identify it in the truststore.
- The script checks whether the certificate is expired or not yet valid. The script won't import an expired or invalid certificate.
- The script validates whether there already is a certificate in the truststore with the same alias. If so, it won't import the certificate.

If the preceding validations are met, the X.509 certificate is imported in the truststore.

List

The script can list the details of all public key certificates in the Application Studio truststore. The details include:

- The alias of the certificate.
- The date when the certificate was imported in the truststore.
- The date when the certificate expires.
- The first two and the last two characters of the SHA-1 fingerprint of the certificate.

Delete

The script can delete public key certificates in the Application Studio truststore.

The script prompts for the alias of the certificate to delete. If a certificate with the alias exists in the truststore, it is deleted.

Private certificate management

The Apache Tomcat web server used by Application Studio has a self-signed certificate for SSL that was generated during installation. Best practice after installing Application Studio is replacing the default certificate with your own certificate. You can use the managePersonalCerts tool to replace it.

You can use the private certificate management tool to import and manage private-public key pairs and certificates in the Application Studio keystore. The supported file format is P12 (.p12). The script, named `managePersonalCerts.sh`, can perform the following functions.

Import

The script can import X.509 private key certificate files.

- The script prompts for the name of the PKCS #12 file containing the key pair and the password set when the certificate was exported.
- The script checks whether the certificate is expired or not yet valid. The script won't import an expired or invalid certificate.

If the preceding validations are met, the X.509 certificate is imported in the keystore. The certificate is given the default alias of the previous certificate in the keystore (that is, **tomcat**). Restart Application Studio for Tomcat to use the certificate.

Note The default installation of Application Studio creates a self-signed certificate and keeps it in the keystore under the default alias **tomcat**. When the `managePersonalCerts` utility is run, it deletes the existing certificate from the keystore and imports the certificate provided by the user, keeping the alias of the imported certificate the same as before. Hence the newly imported certificate has the same alias as the previous one.

List

The script can display the following details about a certificate in the keystore: alias, serial name and validity dates.

Repair database connection 12

Application Studio has a command-line script for repairing its connection to the database. You need this tool if the database has been moved to another server or the credentials for connecting to the database have changed.

The script is `repair.sh` and is in the `<install directory>\cli-tools` directory

Invoke the script by executing `repair.sh datasource`.

The script prompts you to select a database type. Then you can change or keep the current connection settings as needed.

Restart Application Studio after running the script. In the administration user interface, your changes override the datasource settings for the currently selected profile at System Settings > Datasource & Profile Settings.

Set new shared secret 13

Use this procedure if you have forgotten the Application Studio shared secret that was set during installation. The following procedure replaces the current shared secret with a new one. Application Studio uses the shared secret as a key for encrypting and decrypting passwords. Changing the shared secret requires the follow-up task of changing all passwords.

Stop Application Studio. Run the following command and enter a new shared secret when prompted.

```
<install directory>/cli-tools/recover.sh shared_secret
```

Replacing the shared secret invalidates passwords of all users, including the default admin user. You need to reset the admin user's initial password to the factory default value **axway**. This enables the admin user to log on and change passwords of all other users. In the following step **6a5b301829a83d56073ba6b33ee3dda6** is the encrypted value of **axway**.

Connect to the database using an SQL client and run following commands for your database type to reset the admin password to the axway default.

MySQL

```
UPDATE dir_user SET password='6a5b301829a83d56073ba6b33ee3dda6',
active=1 WHERE username='admin';

UPDATE dir_user_extra SET algorithm='', requiredPasswordChange =
true WHERE username='admin';
```

Oracle

```
UPDATE dir_user SET password='6a5b301829a83d56073ba6b33ee3dda6',
active=1 WHERE username='admin';

UPDATE dir_user_extra SET algorithm='', requiredPasswordChange = 1
WHERE username='admin';
```

Then run the following command to rehash the database password.

```
<install directory>/cli-tools/repair.sh datasource
```

Caution This erases all customization in the `datasource` file (for example, Oracle RAC, SLL).

Start Application Studio, log on as the admin user and reset the password when prompted. Now the admin user must reset passwords of all users and passwords for all plugins and form fields.

Enable HSTS

14

You can enable HTTP Strict Transport Security (HSTS) to force browsers to connect securely via TLS or SSL when accessing the Application Studio user interface. Enabling HSTS involves editing an XML file to add some properties.

1. Open the `web.xml` file at `<install directory>/app-studio/apache-tomcat-<version>/webapps/appstudio/WEB-INF`.
2. Search for the `JsonResponseFilter` in the file.
3. Add the following snippet before the `JsonResponseFilter`. This placement is necessary so the HSTS filter processes all requests.

```
<filter>

        <filter-name>HstsFilter</filter-name>
        <filter-
        class>com.axway.defence.servlet.http.HstsFilter</filter-
        class>
        <init-param>

                <param-name>httpsPort</param-name>
                <param-value>443</param-value>

        </init-param>
        <init-param>

                <param-name>maxAge</param-name>
                <param-value>86400</param-value>

        </init-param>
        <init-param>

                <param-name>includeSubdomains</param-name>
                <param-value>false</param-value>

        </init-param>

</filter>
<filter-mapping>

        <filter-name>HstsFilter</filter-name>
        <url-pattern>/*</url-pattern>

</filter-mapping>
```

4. The preceding snippet has default values for `httpsPort`, `maxAge` and `includeSubdomains`. You might have to edit the values for your environment. The following describes the parameters.

   httpsPort

   The port to forward HTTPS traffic.

maxAge

The maximum time in seconds that browsers recognize the HSTS policy.

includeSubdomains

Specifies whether subdomains of the requested domain also should recognize the policy.

5. Save the file and restart Application Studio for the changes to become effective.

Product security 15

This documentation provides instructions and recommendations to help you strengthen the security of Application Studio. Security descriptions include:

- How the product was developed in a secure way
- A list of main security features
- Secure configuration parameters, including the secure by default configuration
- Identity and access management in this product
- Best practices to use this product in a secure way

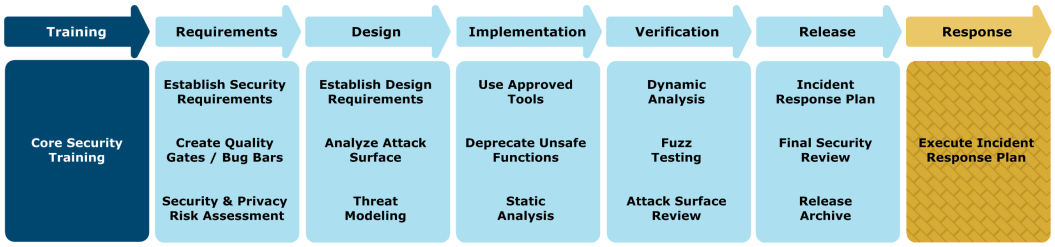
This documentation is intended for:

- Security teams in charge of auditing the security of the product
- Global network engineers
- Product administrators

Secure Development Lifecycle

Axway implements a Secure Development Lifecycle (SDL) in the development of its main products. This SDL is similar to the one defined by Microsoft. A dedicated team, the Product Security Group (PSG), in association with the product development teams, is in charge of managing this SDL.

The secure development lifecycle consists of standardized process flows during the development lifecycle. The SDL process is a spiral development model. Axway works within this model and with an Agile development model to deliver the required security artifacts for the SDL. The following depicts the simplified Microsoft SDL model.



Axway implements many SDL process and control points. Some important steps along the path to secure product delivery include the creation of both security and design requirements as well as the associated design-time threat models. These steps ensure that products meet initial specifications and that the posited security designs pass all the threat model criteria established both within the security community and with the Axway PSG.

At Axway, there is a broad suite of tools associated with the implementation and verification phases of the SDL. Both static and dynamic analysis tools are run to identify potential code weaknesses and to discover security issues potentially exposed at runtime. Suites of attack surface tools and penetration testing tools are run to ensure the deployed products on the target platforms meet gated criteria in the SDL and criteria provided by the internal PSG. Axway provides tool suite information and our customized usage profiles upon customer request. Well-known security industry-standard tools are used as well as many other enhanced test scenarios required by our most security-conscious customers.

The new product introduction (NPI) process at Axway requires a final security review with associated development and testing artifacts. This NPI process supports the release of new products or major product revisions. The NPI also ensures the SDL is started early in development to optimize the delivery of secured products for you, our customer.

Security features

Application Studio has features for enhancing security. These include secure connections and password, certificate and user management. The following is a summary.

Secure connections

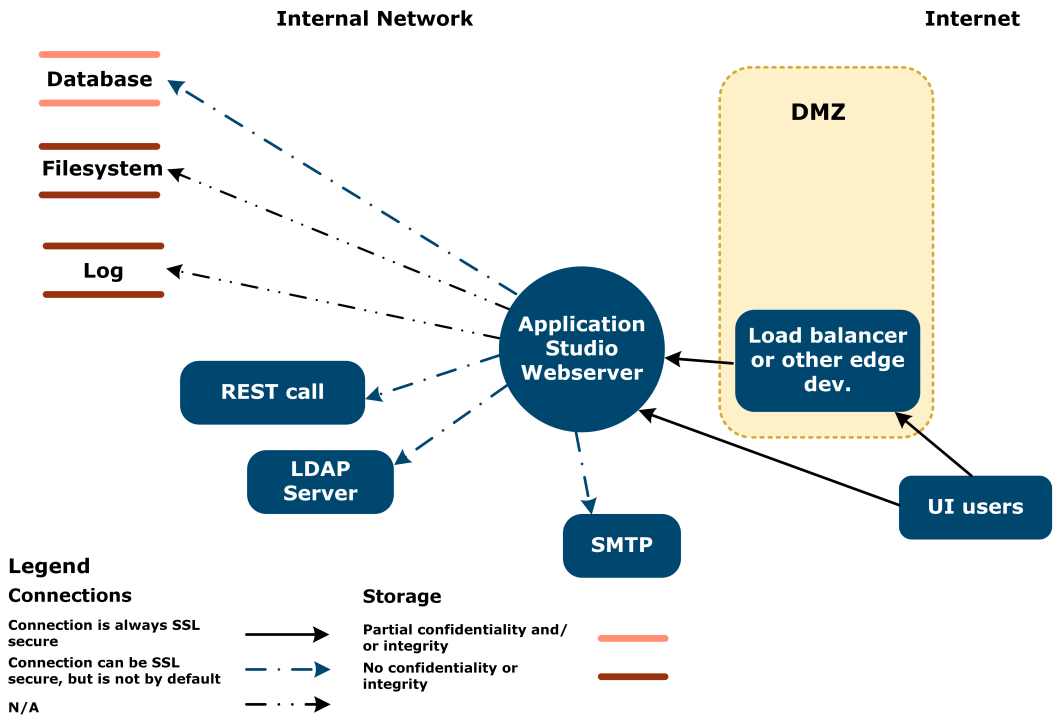
Application Studio supports the following secure connections.

- Connections between users and the Apache Tomcat web server used by Application Studio.

  These include user interface access and Representational State Transfer (REST) application programming interface (API) calls that support JavaScript Object Notation (JSON) format.
- Connections for Simple Mail Transfer Protocol (SMTP) email integration.
- Connections from plugins included in Application Studio.

  See *REST Form Store Binder plugin* on page 16 and *Email plugin with status options* on page 40.

The following graphic illustrates connections in Application Studio.



Password management

Application Studio has the following password features.

- Passwords of Application Studio users are encrypted with HMAC SHA-256 using a combined salt composed from the shared secret and a new salt generated per password.
- Other passwords, such as a database password, are stored encrypted with AES-256. The encryption process uses the shared secret and a new initialization vector.
- The shared secret can be changed with a command-line tool and manual recovery process. See *Set new shared secret* on page 61.

Certificate management

Application Studio has the following certificate management features.

- The Application Studio web server private key and trusted public certificates are stored in Java keystores: `.appstudio-keystore` and `.appstudio-truststore`. These are at `<installation user's home>/.appStudio`.
- The keystore is secured by a password generated during installation. The password is used in `<installation directory>/apache-tomcat-[version]/conf/server.xml`. See *Sensitive files and databases* on page 76 for information about safeguarding this file.

- Certificates are managed using two scripts: `managePersonalCerts.sh` and `manageTrustedCerts.sh` at `<installation directory>/cli-tools`. See *Certificate management tools* on page 58 for details.
- The self-signed server certificate created during installation is intended for testing purposes only. Best practice is for users to obtain and import their own signed certificate. See *Server certificate* on page 74 for more information.

  When you replace the certificate with your own, note that the certificate revocation list and certificate path are not checked.

User management

The following features are available for managing users.

- Role-based access control (RBAC) is used. Available roles are Admin and User. Only users with the Admin role can use the workflow designer.
- RBAC for workflow process and form activities can be further controlled by user roles represented by groups, departments, organizations and so on.
- By default users are stored in the local database. Alternately, user management can be integrated with an external Lightweight Directory Access Protocol (LDAP) server. See *LDAP user authentication* on page 50.

Identity and access management

Application Studio implements a role-based access control (RBAC) model for identity and access management (IAM). For the application designer user interface, available roles are Admin and User.

The designer interface provides great flexibility on how RBAC is implemented for each application developed in Application Studio. Access to processes, forms, user views and even subsections of those can be defined per user or by a user's membership in different user-management objects.

See the Joget Workflow knowledge base for specific user-management objects and the application designer user interface for permissions that use them. The URL for the KB is:

http://dev.joget.org/community/display/KBv4/Setup+Users

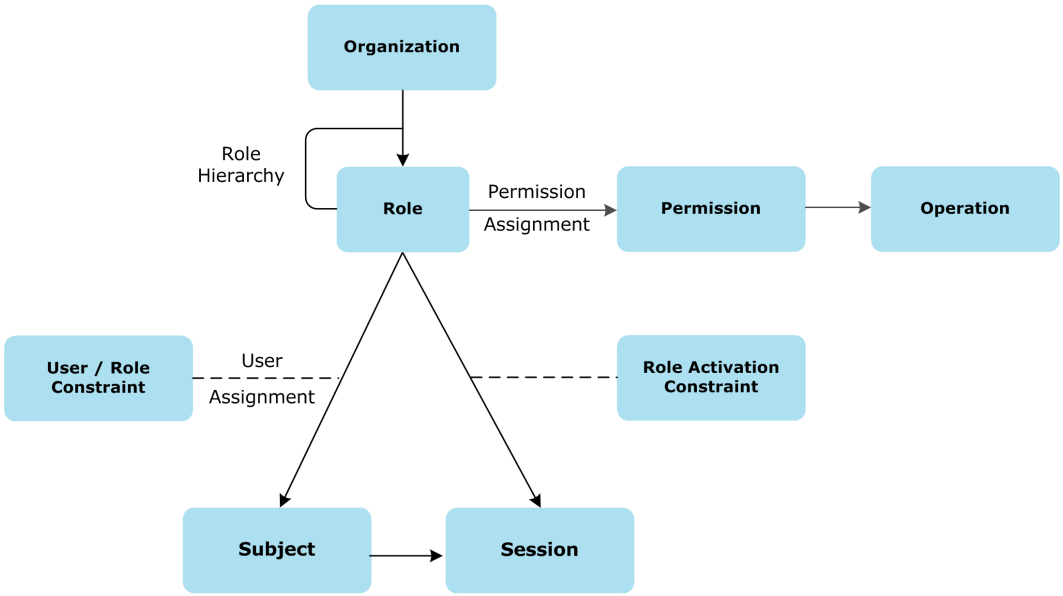
Assigning users to groups created to represent a role or access, and then requiring that group permission to perform workflow, form or user view activities, is a common way to implement RBAC on created applications. As illustrated in the generic RBAC model:

- Role can be understood as a group
- Permission can be understood as implemented in the designer for appropriate application activities.

RBAC model

This RBAC model has the concept of few roles within an organization. Instead of assigning access to each individual resource, permissions can be assigned to roles, which users are assigned to. This makes it easier to manage permissions. You do not have to review the complete list of users when a new resource is added, but only review the list of roles. Similarly, you do not have to review the complete list of resources when a user is added, but only review the list of roles.

The following diagram illustrates a generic RBAC model.



The following describes how the RBAC model works.

- User membership in a group, department or organization is a used for permissions as part of workflow application design.
- A permission is the right to access or perform actions in a workflow or application. For example, a permission defined in the application designer on a workflow process, form or userview could be:

  is admin
  logged in user
  organization
  department
  group

Uniform Resource Identifier (URI) resources are protected with a configuration based on the roles. You can add URLs under the XML element `security:http / security:intercept-url` in the file `<installation directory>/apache-tomcat-<version>/webapps/appstudio/WEB-INF/classes/customApplicationContext.xml`. Restart Application Studio for changes to become effective.

Available IAM resources

The following are available IAM resources in Application Studio.

- Workflow applications support fine-grained access control (FGAC) of components and activities by users. Objects subject to FGAC are organizations, departments, groups and users. For example, you can specify that a user can view or change only certain forms. Moreover, permissions can be enforced for multiple users through use of organizations, departments or groups.

  While Application Studio applications support requiring permissions on various components, that feature must be used during application design. Best practice is to consider appropriate user access controls at the beginning and design them into each of your applications.

- The following describes the possible roles in Application Studio.

  Admin

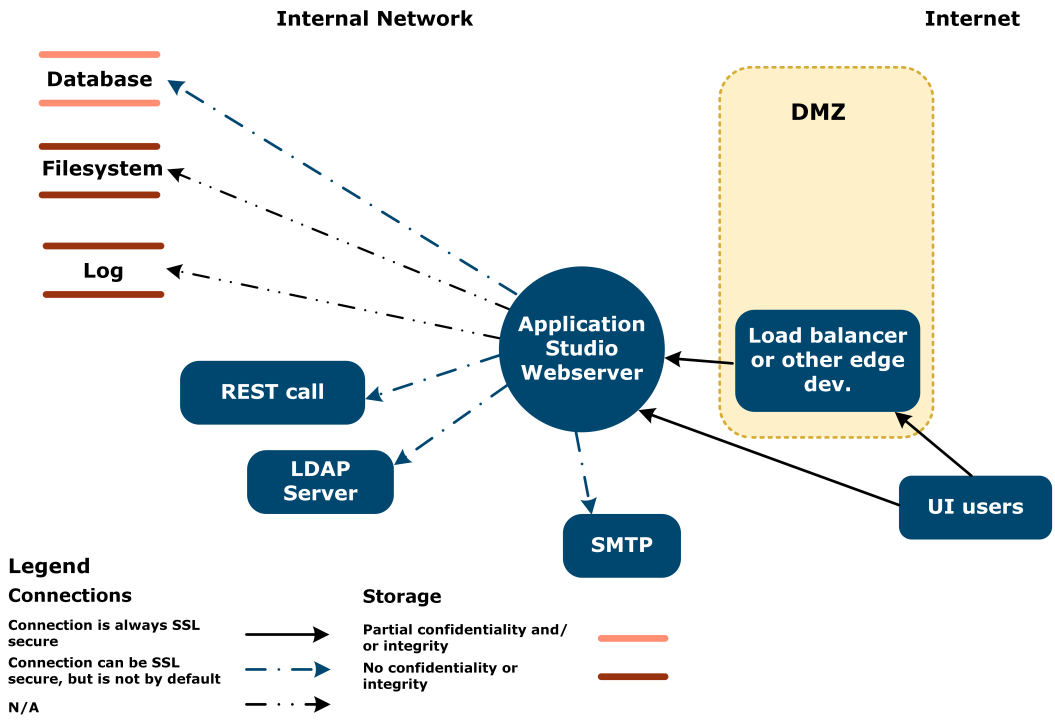
  This role enables access to appstudio, web/console/home and related pages. Users with this role have unlimited authority to perform all tasks in the designer. Users given the admin role can administer applications, system settings, manage users and groups, and design and deploy applications.

  User

  Users with the user role do not have access to administrator activities, but can access published applications if the application's specific permissions allow.

Security architecture

The following diagram provides an overview of the network and storage connections in an implementation of Application Studio.



The diagram includes the following components:

- The designer console and workflow application user interface runs on an Apache Tomcat web server. Users connect to the UI from a web browser via HTTPS. By default any HTTP UI connection is redirected to the HTTPS port.
- A JSON REST API permitting management of workflow processes is part of the UI. The JSON REST API supports passing credentials using HTTP basic authentication.
- Integration-side connections like SMTP and REST can be secured via Transport Layer Security (TLS).
- Connection to an external LDAP server can use TLS.

Security configuration

The following topics describe security configuration for Application Studio.

Inbound SSL configuration

Application Studio supports HTTPS connections by default. HTTP connections are redirected to the HTTPS port.

Outbound SSL configuration

Application Studio can use Secure Sockets Layer (SSL) connections to back-end services like SMTP and REST. See the user interface configuration for enabling or disabling TLS for each of these connections.

Host name verification as part of the TLS handshake for SMTP and REST client connections can be found at:

```
<installation directory>/apache-
tomcat<version>/webapps/appstudio/WEB-
INF/classes/customApplicationContext.xml
```

Key properties:

```
Under bean id=”SMTPProperties“, property name=”validateCertificate”
(defaults to true)

Under bean id=”restClient”, property name=”verifyHostname”,
(defaults to false)
```

Certificates and keys in the trust and key stores

Certificates and the server private key can be imported to the trust and key stores and used to secure inbound and outbound TLS connections.

For more information see *Certificate management tools* on page 58.

Password policy

You can configure a password policy for users in Application Studio. You can configure the following rules for user passwords:

- Failed login attempts and lockout period
- Require change on first log on
- Enable forgot password feature and validity period for forgot password link on the log-on page
- Password validity period
- Number of unique passwords before reuse
- Restrictions on password length
- The minimum number of upper-case, lower-case, numeric and special characters a password must contain
- Restrictions on account user name in password

For more information see the Application Studio Developer's Guide.

Shared secret

When installing Application Studio you are prompted to set a shared secret. It is used for generating a common deployment salt and is part of the process for encrypting user passwords and the database password.

Document and save your shared secret in a safe place, such as in a secure password manager. The shared secret is shared between the database and one or more instances of Application Studio.

Administrator password change

After installing Application Studio, the administrative user (admin) must change the password of the admin account from the initial value (axway).

Apache Tomcat security

The configuration for the inbound HTTPS connections for Apache Tomcat is in its configuration file:

```
<installation directory>/apache-tomcat<version>/conf/server.xml
```

The following is a partial list of attributes you can customize:

- sslEnabledProtocols – You can set enabled SSL/TLS protocols.
- ciphers – You can restrict the list of ciphers to support SSL/TLS connections.
- keystoreFile – The path to the file that contains the server certificate with the private key.
- keystorePass – The password used to protect the keystore
- keystoreType – Format of the keystore. If not specified, the keystore is in Java KeyStore (JKS) format.
- keyAlias – Specifies which private key entry to use from the keystore. If not supplied, it takes the first one from the keystore.
- keyPass – The password used to protect the key inside the keystore. If not supplied, the value from keystorePass is used.
- useServerCipherSuitesOrder – enforce the server's preferred cipher order rather than accept the first mutually agreeable one presented by the client.

Default cipher suites

Application Studio by default has a list of TLS 1.2 ciphers that are generally considered secure. If support of older browsers is needed you might have to add additional TLS 1.0 ciphers to the ciphers list. Add them second to last at the end of the list (between the existing ciphers and the special TLS_EMPTY_RENEGOTIATION_INFO_SCSV) so that **useServerCipherSuitesOrder** ensures that each client connection uses the best cipher it can. Furthermore, Axway recommends only adding the least number necessary and the most secure of these older ciphers.

See *Default cipher suites* on page 77 for a list of cipher suites Application Studio supports.

Caution

Changes to Tomcat configuration files like `server.xml` can affect security significantly. Use caution when making changes and maintain controls on who is able to access and change these files.

Also, the latest versions of TLS, 1.1 and 1.2, must be enabled in older browsers as required in the browser security settings to allow browsers to use the latest versions.

Additional information

Additional information about securing Tomcat is at:

https://tomcat.apache.org/tomcat-7.0-doc/security-howto.html

https://www.owasp.org/index.php/Securing_tomcat

LDAP configuration

An external LDAP server is supported. Certain features related to internal user account management may not be available for LDAP-managed accounts.

See *LDAP user authentication* on page 50 for more information.

Local database connection

The database connection is by default configured in:

```
<installation directory>/wflow/app_datasource.properties and app_
datasource-<source name>.properties.
```

The database password is stored encrypted. To change the datasource, use the Designer UI option System Settings – Datasource & Profile or the command line tool `<installation directory>/cli-tools/repair.sh`.

Note that changes to the datasource may affect licensing.

See *Repair database connection* on page 60 for more information about `repair.sh`.

System licensing

See the installation guide for information on system licensing, but be aware that changes to network configuration (number of adapters or adapter order), MAC address or database configuration may require you to re-license Application Studio.

Secure by default configuration

Application Studio is secured by default after initial setup and configuration. You need to explicitly import trusted certificates for integration connections like SMTP or REST servers. The default configuration for TLS protocols and ciphers is secure and should not be changed unless specifically instructed. Currently, Application Studio supports TLS 1.2, TLS 1.1 and TLS 1 protocols, with a preconfigured list of generally considered secure TLS 1.2 ciphers. SSL 2 and SSL 3 protocols are deemed insecure and are not enabled by default.

Although activated by default, the installer-created self-signed certificate used for securing the Tomcat server for TLS communication is intended for testing and should not be used in production. The certificate must be replaced with your own certificate as soon as possible.

During the installation of Tomcat, the HTTP port is configured to redirect to HTTPS, and HTTPS is enabled by default. Axway does not recommend enabling web access to HTTP for other than redirecting to the secure web pages.

The SSL 3 protocol is disabled by default in Java 8. For more information see the Java 8 documentation.

The following measures have been taken to ensure Application Studio is secure by default:

- The default behavior of the password policy is to force users to set a strong password on first log on, including the admin user.
- User passwords are stored as a salted hash using 100000 iterations of Password-Based Key Derivation Function 2 (PBKDF2) with HMAC SHA-256.
- HTTPS interfaces are configured with secure cipher suites by default.

Security best practices

The following best practices are recommended for Application Studio.

Secure connections

Secure connections with external networks are required, but are good for internal connections, too. For example, connections between a REST server and an application must be TLS-secured.

Server certificate

A self-signed SSL server certificate was created during installation. Use it for test purposes only.

Caution Axway strongly recommends using your own certificate when Application Studio begins managing real data or the product is being used in a production environment.

To change the server certificate:

1. Obtain a server certificate. The procedure to do this depends on the issuer of the certificate and is outside the scope of this documentation.
2. Use `<installation directory>/cli-tools/managePersonalCerts.sh` to import the new certificate
3. Restart the Application Studio server.

For more details see *Private certificate management* on page 59.

Privileged access user list

Take care to maintain and limit the users who have privileged access to the computer on which Application Studio is running.

Maintain a list of users with privileged access. At minimum, maintain a list of users with the admin role and a list of users with operating system access to the installation directory structure.

Internet access

Limit the number of Internet access points to the extent possible. Allow only necessary Internet connections. Limit interconnections with external networks to the extent possible. This reduces the risk of external attacks and makes it easier to audit the product. If all applications developed in Application Studio are intended for internal users, access could be limited to the internal network.

Update procedure

In the event of a possible vulnerability discovered in Application Studio, you must be able to apply an update as soon as possible. Make sure you have the correct procedure to complete the update. Always use the latest version, if possible, as it contains fixes to known vulnerabilities.

Generic or anonymous users

Use of generic or anonymous users is not recommended. These are users whose passwords are shared by multiple people. This makes it easier for an attacker to retrieve the passwords. In addition, the procedure to change shared passwords can be complex and risky. Such users make it impossible to determine who completed erroneous actions.

Password policy

The password policy governs the size and complexity of passwords and the rules for managing them. The size and complexity are important. The policy should define that the length be a minimum of 8 characters, contain a mix of alphabetic characters with numbers and special characters and be case-sensitive.

Your password policy:

• Must force passwords to be changed periodically.

• Should prohibit reuse of a password before a specified number of different passwords have been used and within a certain interval.

• Must limit the number of failed attempts.

Application Studio stores passwords for connecting to databases, SMTP servers, REST servers, and so on. In these cases, the administrator of the third-party product is responsible for implementing the password policy, and the Application Studio administrator is responsible for keeping these locally configured passwords up to date.

Default authentication account

Application Studio is delivered with the following default administrative user:

- User: admin
- Password: axway

You must change the password after logging on with this user the first time. Creating another account with the admin role is recommended. However, deleting the default admin user is not recommended. It could be important if Application Studio needs to be re-licensed or have its shared secret reset.

Logging, audit and alert rules

You must define correct levels of event logging and auditing to be alerted to errors to research and resolve. The admin console has a page that allows you to view logs. Experienced users can modify log levels by editing the following file:

```
<installation directory>/apache-
tomcat<version>/webapps/appstudio/WEB-INF/classes/log4j2.xml
```

Sensitive files and databases

You must protect sensitive files and databases. The configuration file contains information that can be useful to a hacker. Even if part of this information is encrypted, access to this file must be restricted by your local access management. Only the product runtime user and administrators should have access to this file in any mode (read, update, delete).

The file containing the shared secret used to generate encryption keys must be protected as well. The contents are encrypted, but access to this file should only be granted to the product: `<user home>/.appStudio/.appstudio`.

You must protect the files containing the server private key and trusted certificates. Though the keystores are secured by a password that is randomly generated at installation, that password is used in `conf/server.xml`. Access to these files should only be granted to the product runtime user: `<user home>/.appStudio/.appstudio-keystore` and `.appstudio-truststore`.

The Tomcat `server.xml` file contains important security settings and should be protected. It has web server and TLS settings as well as the keystore password: `<installation directory>/apache-tomcat-[version]/conf/server.xml`.

The following is a partial list of other sensitive files and directories:

- `<user home>/.appStudio/`
- `<installation directory>/wflow`
- `<installation directory>/cli-tools`
- `<installation directory>/apache-tomcat<version>/conf`
- `bin`, `lib`, `logs` and `classes` directories under the `apache-tomcat` directory.

The database also contains sensitive data. Prohibit access to the database, except for administrators.

Also protect any custom logging files if a non-default location has been configured.

Strong protocols and ciphers

Avoid protocols and ciphers regarded as insecure under today's industry standards. At the time of this writing SSL 3 does not provide enough security and should not be used. Instead, only use TLS 1.2 (preferred), TLS 1.1 or TLS 1. Application Studio comes configured with TLS 1.2 ciphers considered sufficiently strong. Use caution when adding lower strength ciphers for backward compatibility.

Default cipher suites

The following are the default cipher suites Application Studio supports. These are from the `server.xml` file at `<installation directory>/apache-tomcat<version>/conf`.

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P384

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P256

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_DSS_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

TLS_DHE_DSS_WITH_AES_256_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P384

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P384

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_DSS_WITH_AES_128_GCM_SHA256

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256

TLS_DHE_DSS_WITH_AES_128_CBC_SHA256

TLS_DH_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_RSA_WITH_AES_128_GCM_SHA256

TLS_DH_DSS_WITH_AES_256_GCM_SHA384

TLS_DH_RSA_WITH_AES_256_CBC_SHA256

TLS_DH_DSS_WITH_AES_256_CBC_SHA256

TLS_DH_DSS_WITH_AES_128_GCM_SHA256

TLS_DH_RSA_WITH_AES_128_CBC_SHA256

TLS_DH_DSS_WITH_AES_128_CBC_SHA256

TLS_EMPTY_RENEGOTIATION_INFO_SCSV

Glossary

active-active

Active-active is a type of high-availability cluster environment. In an active-active cluster, traffic intended for the failed node is forwarded to an existing node or load-balanced across the remaining nodes. This is usually possible only when the nodes have a homogeneous software configuration.

AES

Advanced Encryption Standard (AES) is one of the most frequently used and most secure encryption algorithms available today.

Ant

Apache Ant is a software tool for automating software build processes. Ant stands for "another neat tool."

API

An application programming interface (API) is a protocol intended for use as an interface by software components to communicate with each other.

BPM

Business process management (BPM) is the discipline of managing processes for improving business performance outcomes and operational agility. Processes span organizational boundaries, linking people, information flows, systems and other assets to create and deliver value to customers and constituents.

CA

A certificate authority or certification authority (CA) is an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified. A CA is a third party trusted by the subject (owner) of the certificate and the party relying upon the certificate.

cipher suites

A cipher suite is a set of cryptographic algorithms used for the following: Protect information required to create shared keys (key exchange), encrypt messages exchanged between clients and servers (bulk encryption), generate message hashes and signatures to ensure the integrity of a message (message authentication).

CSV
A comma-separated values (CSV) file stores tabular data (numbers and text) in plain-text form. Also sometimes called character-separated values because the separator character does not have to be a comma.

datalist
An object for retrieving data in the database or data from another source, such as a CSV file.

dataset
A row of data in a database table

DBA
database administrator

FGAC
Fine-grained access control (FGAC) is a way to manage users' access to objects or capacity to perform actions. For example, you could enable some users to view specific objects in the user interface, but prohibit other users from viewing the same objects.

HMAC
In cryptography, a keyed-hash message authentication code (HMAC) is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key.

HMAC SHA-256
The HMAC SHA-256 class computes a hash-based message authentication code (HMAC) by using the SHA-256 hash function.

HSTS
HTTP Strict Transport Security (HSTS) is a web security policy for protecting websites against protocol downgrade attacks and cookie hijacking.

HTTP
HyperText Transfer Protocol (HTTP) is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions web servers and browsers should take in response to various commands.

HTTP basic authentication
With HTTP basic authentication the user agent must authenticate itself with a user-ID and a password for each realm. HTTP basic authentication is the simplest technique for enforcing access controls to web resources. It does not require cookies, session identifier and login pages. Rather, it uses static, standard HTTP headers, which means no handshakes have to be done in anticipation.

HTTPS

HTTPS is a protocol for secure communication over a computer network widely used on the Internet. HTTPS consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security or its predecessor, Secure Sockets Layer. HTTPS is also called HTTP over TLS, HTTP over and HTTP Secure.

IAM

Identity and access management (IAM) is a role-based solution for securing enterprise resources and managing user access to protected network components through a continuous and interactive authorization process.

J2EE

Java 2 Platform Enterprise Edition (J2EE) is a platform-independent, Java-centric environment for developing, building and deploying web-based enterprise applications online. The J2EE platform consists of a set of services, APIs and protocols that provide the functionality for developing multi-tiered, web-based applications.

JDK

The Java Development Kit (JDK) is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, Mac OS X or Windows.

JKS

A Java KeyStore (JKS) is a repository of security certificates – either authorization certificates or public key certificates – used for instance in encryption.

JSESSIONID

JSESSIONID is a cookie generated by a servlet container like Tomcat or Jetty and used for session management in J2EE web applications for the HTTP protocol.

JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format. JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

JSONPath

A specification syntax for accessing elements of a JSON object.

LDAP

The Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry-standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network.

list grid
A table of data that comes from a datalist.

lists
Lists are a means to retrieve submitted form data and for reporting purposes. Lists often are used to complement processes and forms.

MAC address
A media access control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are used as a network address for most IEEE 802 network technologies, including Ethernet.

NAS
Network-attached storage (NAS) is file-level computer data storage connected to a computer network providing data access to a heterogeneous group of clients. NAS operates as a file server. It is specialized for this task by its hardware, software or configuration of those elements.

NFS
Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems in 1984. It allows a user on a client computer to access files over a network in a manner similar to how local storage is accessed.

PBKDF2
Password-Based Key Derivation Function 2 (PBKDF2) is a key derivation function that is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898.

RAC
Oracle Real Application Clusters (RAC) provides software for clustering and high availability in Oracle database environments.

RAID
Redundant array of independent disks (RAID) is a storage technology that combines multiple disk drive components into a logical unit for the purposes of data redundancy and performance improvement. Data are distributed across the drives in one of several ways, referred to as RAID levels, depending on the specific level of redundancy and performance required.

RBAC
In computer systems security, role-based access control (RBAC) is an approach to restricting system access to authorized users. RBAC is sometimes referred to as role-based security.

REST
Representational State Transfer (REST) is an architecture often used in the development of web services. It incorporates the simple, stateless use of the GET, POST, PUT and DELETE actions,

over HTTP/S. The use of REST is often preferred for Internet use over the more heavyweight SOAP (Simple Object Access Protocol) because REST does not consume as much bandwidth.

salt

In cryptography, a salt is random data that are used as an additional input to a one-way function that hashes a password or passphrase. The primary function of salts is to defend against dictionary attacks and pre-computed rainbow table attacks. A new salt is randomly generated for each password. In a typical setting, the salt and the password are concatenated and processed with a cryptographic hash function, and the resulting output (but not the original password) is stored with the salt in a database. Hashing allows for later authentication while defending against compromise of the plaintext password in the event that the database is somehow compromised.

SDK

A software development kit (SDK or "devkit") is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.

self-signed certificates

In cryptography and computer security, a self-signed certificate is an identity certificate that is signed by the same entity whose identity it certifies. This term has nothing to do with the identity of the person or organization that actually performed the signing procedure. In technical terms a self-signed certificate is one signed with its own private key.

SHA-1

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST.

SHA-256

The SHA (Secure Hash Algorithm) is one of a number of cryptographic hash functions. A cryptographic hash is like a signature for a text or a data file. SHA-256 is one of the successor hash functions to SHA-1, and is one of the strongest hash functions available.

SMTP

Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (email) transmission.

SSL

Secure Sockets Layer (SSL), which is the predecessor of Transport Layer Security (TLS), is an encryption protocol that ensures communication security over the Internet. See TLS for more information.

SSL 2.0
SSL 2.0 is Secure Socket Layer version 2.0.

sticky sessions
A sticky session is a feature of many commercial load balancing solutions for web-farms to route requests for a session to the same physical machine that serviced the first request for that session. This is mainly used to ensure an in-proc session is not lost as a result of requests for a session being routed to different servers. Since requests for a user are always routed to the same machine that first served the request for that session, sticky sessions can cause uneven load distribution across servers.

TLS
Transport Layer Security (TLS) is an encryption protocol that ensures communication security over the Internet. TLS encrypts the network connection above the transport layer. TLS uses asymmetric cryptography for key exchange, symmetric encryption for privacy and message authentication codes for message integrity. Secure Sockets Layer (SSL) is the predecessor of TLS.

Tomcat
Apache Tomcat, often referred to as Tomcat, is an open-source web server and servlet container developed by the Apache Software Foundation (ASF).

UFM
see Unified Flow Management

Unified Flow Management
Unified Flow Management (UFM) is a set of products in the Axway 5 Suite that enable you to manage the flow of data within and outside your enterprise.

URI
A Uniform Resource Identifier (URI) is a string of characters used to identify a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web.

userview
A userview is the interface for users to interact with the system. A userview acts as a wrapper for processes, forms and lists.

WAR
A Web application ARchive (WAR) is a JAR file used to distribute a collection of JavaServer Pages, Java servlets, Java classes, XML files, tag libraries, static web pages (HTML and related files) and other resources that together constitute a web application.

workflow

A workflow is an orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes that provide services or process information. A sequence of operations performed by persons or systems.

X.509

In cryptography, X.509 is an ITU-T standard for a public key infrastructure (PKI) and Privilege Management Infrastructure (PMI). X.509 specifies standard formats for public key certificates, certificate revocation lists, attribute certificates and a certification path validation algorithm.