ApplinX User Guide

Designing and Developing an Application

Version 10.11

October 2021

WEBMETHODS

This document applies to ApplinX Version 10.11 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: APX-UG-DESDEVAPP-1011-20211002

Table of Contents

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **https://documentation.softwareag.com**.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.aspx** and give us a call.

Software AG Tech Community

You can find documentation and other technical information on the Software AG Tech Community website at **https://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Developing an ApplinX Application

2 Managing an Application

An application is a project which can contain an internal system within an organization, such as human resources, billing, customer services etc. The following topics provide detailed step-by-step instructions for various application connected tasks:

Refer to Application Configuration Parameters for further details regarding all the application parameters.

Creating an Application

In this task, you will create a new ApplinX application. Every ApplinX application is connected to a host and has a repository containing all the ApplinX entities. The steps detailed in this task include the basic steps required to create an application. Advanced configuration is explained in the reference to Application Parameters.

To create an ApplinX application

Before creating a new application, ensure that ApplinX server is available and running, and start the Designer. Login to the relevant server or add an additional server as required (refer to Designer to the Logging on and Adding a New Server Connection sections.

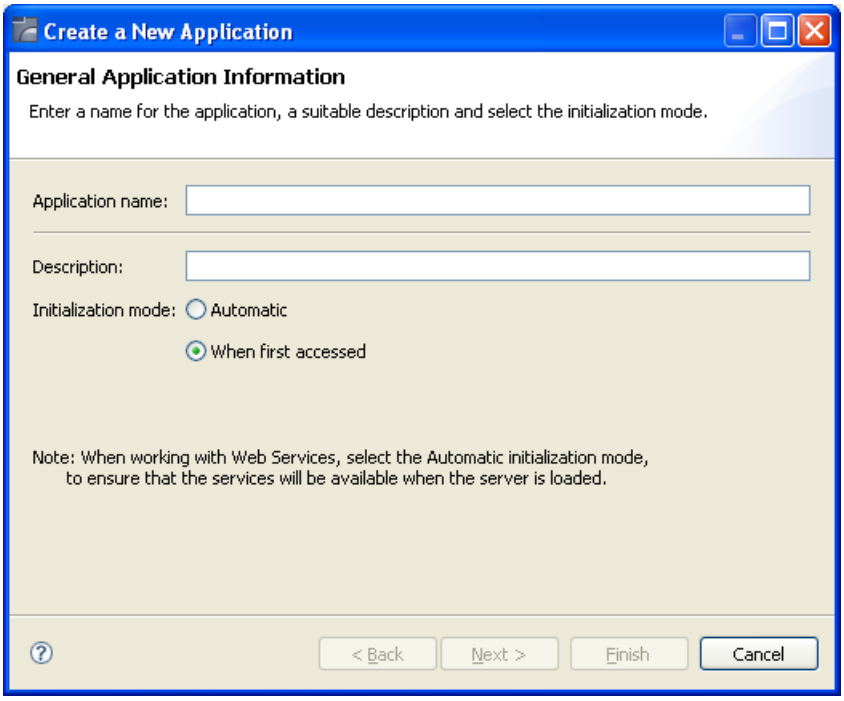
1 In the ApplinX Explorer, right-click on the ApplinX Server and choose **New Application...** .

Or:

Choose the menu item **File > New > Other...**. The *New Project wizard window* is displayed.

Expand the **Software AG** node and choose **Application** then click **Next** to display the New ApplinX application wizard:

Create a New Application wizard is displayed.



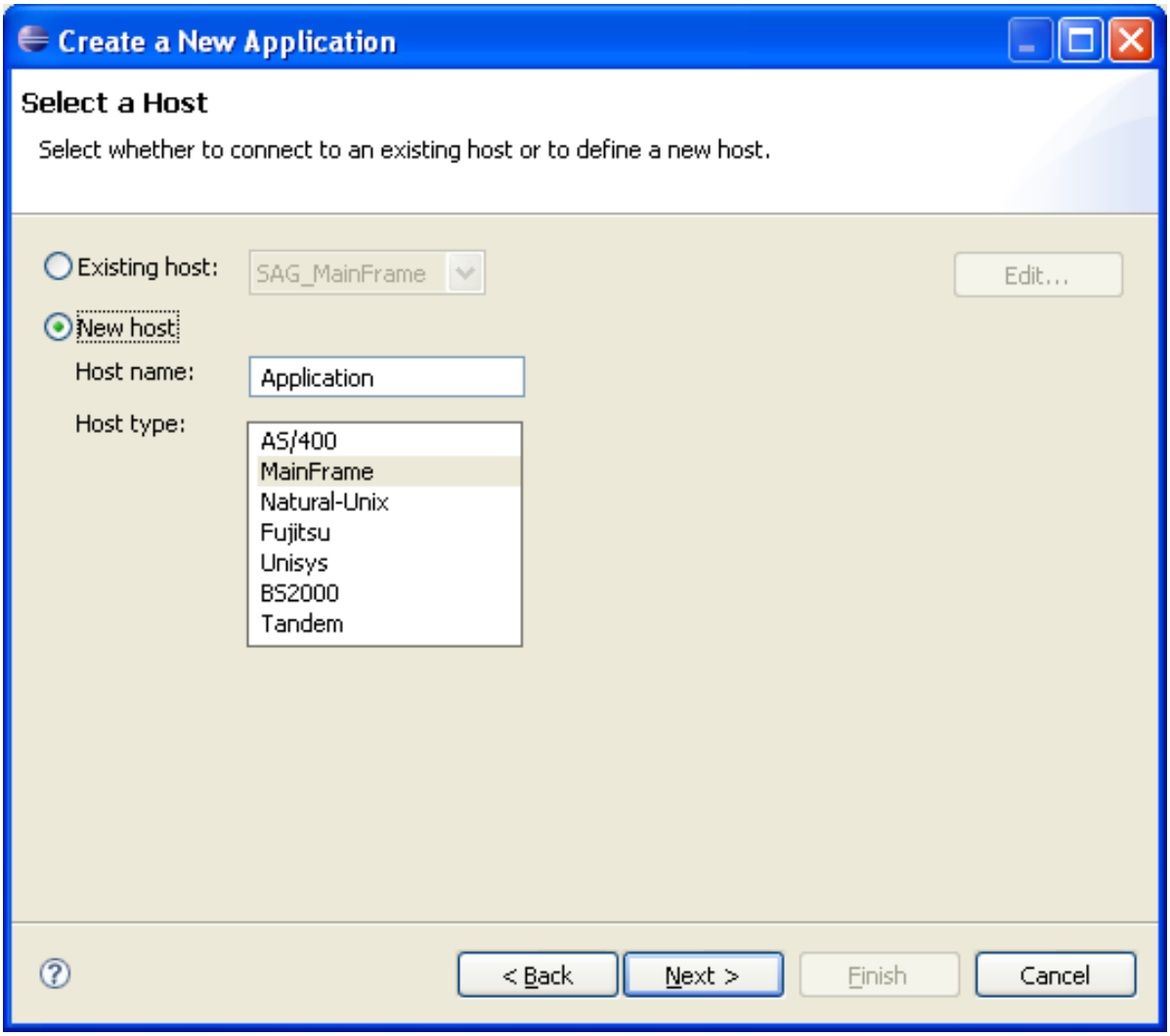
2 Enter a name for the application and a suitable description.

3 Select the **Initialization mode**:

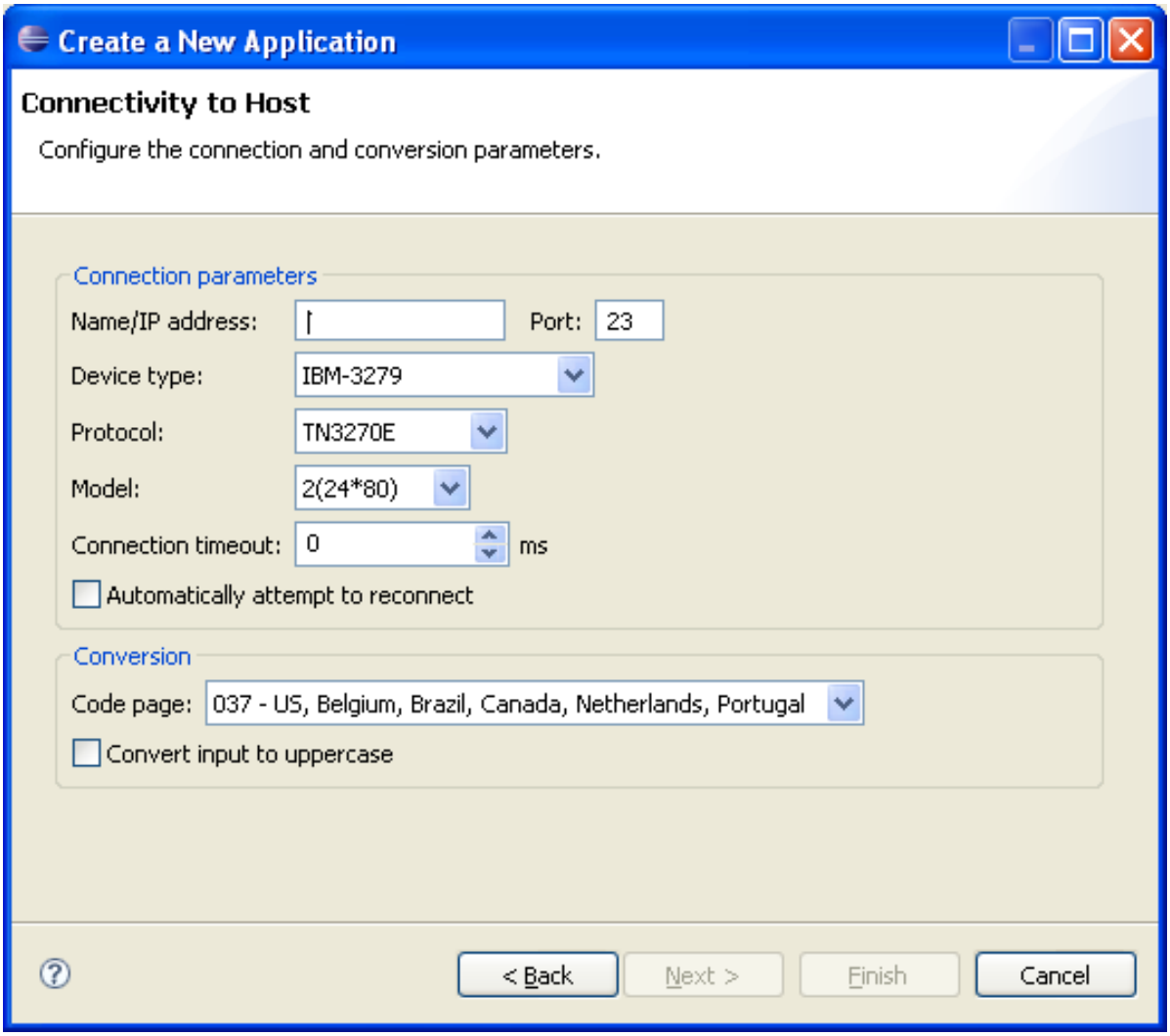
Automatic
Automatically loaded when the server is started.

When first accessed
Loaded when first accessed, in other words, when the code that initializes startup is first called (default).

4 Click **Next**. The *Select Host* screen is displayed. Select whether to use an existing host or to create a new host.



5 When selecting to define a new host, enter a name for the host and select the relevant type of host. Click **Next**. The host connection and conversion parameters are displayed.



It is mandatory to enter the host's name/IP address (IPv4 and IPv6 address formats are supported). Configure the other parameters as required. Further details of these parameters are available in Host Parameters.

6 Click **Finish**. The application is displayed in the ApplinX Explorer. It is possible to further configure the application properties by right-clicking on the application name and selecting **Properties**. Refer to the Application Configuration Parameters in the Reference section for details regarding the advanced configuration properties.

Note: ApplinX provides predefined, built-in transformations. These are a basic kit of commonly used transformations that can be used as-is, or you customize them to suit your exact needs, building up your own transformation library. See Transformations.

To use ApplinX predefined built-in transformations

- Right-click on the Repository node of your selected application and choose **Import Predefined Transformations**.

Editing an Application's Configuration (Advanced Configuration)

The *Application Properties* dialog box contains additional parameters that can be configured such as the language, printer configuration, process tracing, screen content definitions etc. For details of each node of the application properties refer to the Application Reference section. To edit an application's configuration, right-click on the application's node (in the ApplinX Explorer) and choose **Properties**.

Note: Notice that in the **Host name** field (in the Host tab), only hosts of the same type as the configured host are listed.

Deleting an Application

To delete an application

- Right-click on the application's node and choose **Delete**. In addition to deleting the application it is also possible to determine whether when the host used in this application is not used in any other application, you would like to delete the host and also whether you would like to delete the entire application from the file system.

Reloading an Application

Reload an application when there is new data that the application requires, such as when repository definitions change or when the database file is replaced. Reloading the application will reload the repository, close all connection pools, stop running connection pools and if the database is synchronized at the end of the reload process will automatically restart the connection pools.

To reload an application

- Right-click on the application's node and choose **Reload Application**.

Importing an Application's Configuration or Entities

It is possible to import:

- **A complete ApplinX application**, including the application configuration, the application entities and a trace file, i.e. a gxar file.
- **An application's entities**, i.e. a gxz file.

Note: It is not possible to import entities exported from a higher server or database version than the current version that you are using.

Note: In ApplinX versions prior to version 8.0, gxz files included the application's configuration and/or the application's entities. From version 8.0, gxz files include the entities only, and gxar files can include the application's configuration, the entities, and trace files. gxz files from previous ApplinX versions which include the application configuration and you would like to use, require conversion. Refer to Convert Utility Batch File on details regarding converting a gxz file to a gxar file.

Important: As the import process may consume a lot of memory, it is recommended to restart ApplinX server after completing the process.

Importing a Complete ApplinX Application

When importing a complete ApplinX application, you will require a gxar (ApplinX application archive) file. This file includes the application configuration, ApplinX entities (as a read only gxz file) and a trace file. The gxar file can be imported into ApplinX in two different methods:

- "Hot Deploy": The gxar file is simply located and placed in the *host-applications* directory as the file to be imported.
- Import wizard

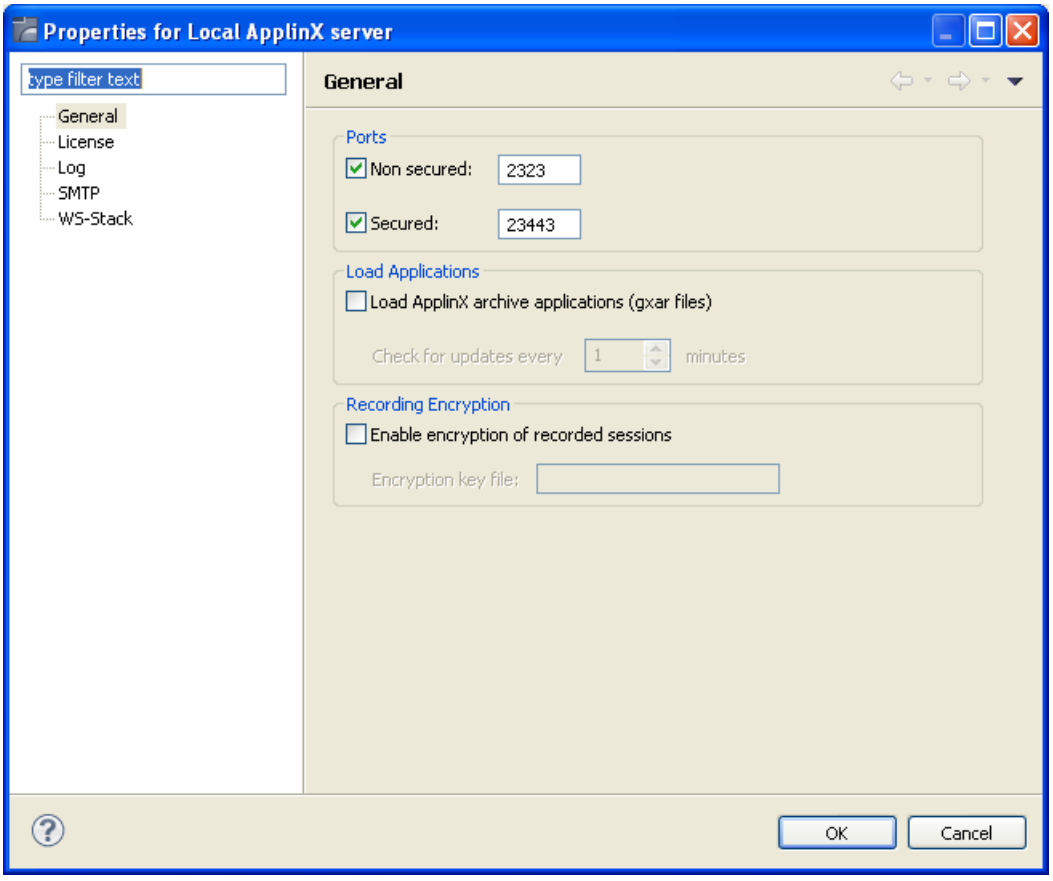
"Hot Deploy"

To import an application in the "Hot Deploy" method

1 Open your Windows Explorer.

2 Locate the relevant gxar file and place it in the *host-applications* directory of your ApplinX installation.

3 In ApplinX Designer, right-click on the server and choose **Properties.**



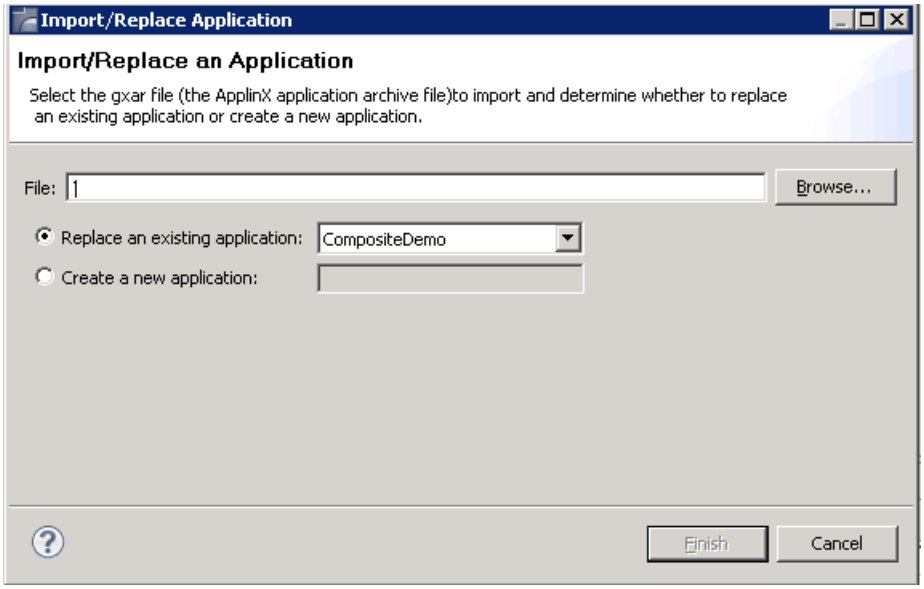
4 In the **General** tab, determine to load ApplinX archive applications. Set the interval to look for updates. This determines how often the ApplinX server should check to see if the gxar file to use has changed. Once a different gxar file is detected, it will be used.

It is also possible to use the gxar file by reloading the server (right-click on the server and choose **Reload**).

Import Wizard

To import an application

1 Right-click on the application's node and choose **Import/Replace Application**. The Import Application wizard is displayed.



2 Enter a file name, or browse and select the file to import.

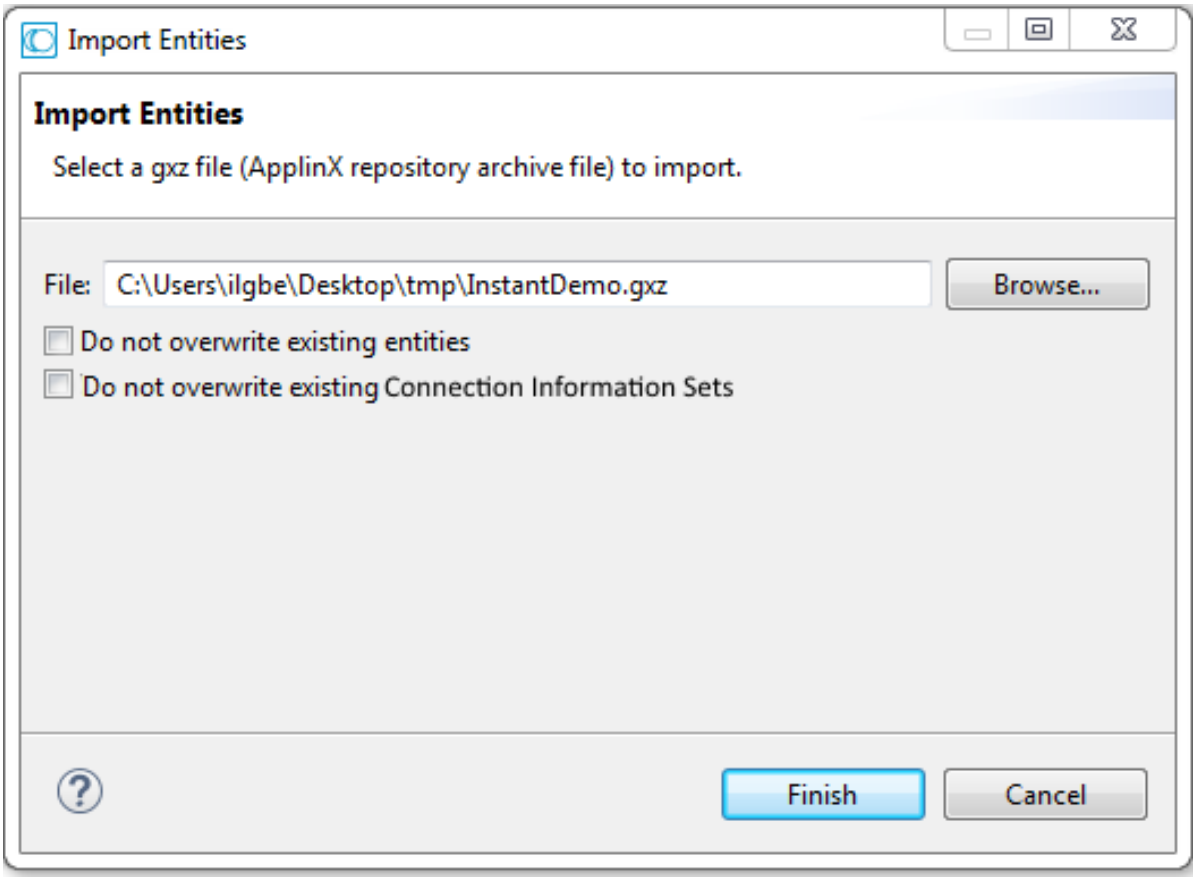
3 Select whether to replace an existing application (selected by default), or to create a new application. Enter a name accordingly.

4 Click **Finish**. Wait a few minutes while the application is imported.

Importing an Application's Entities

To import entities

1 Right-click on the application's Root node and choose **Import Entities**. The **Import Entities** wizard is displayed.



2 Enter a file name, or browse and select the file to import.

Check **Do not overwrite existing entities** to determine not to overwrite an existing entity with an imported entity of the same name.

Check **Do not overwrite existing Connection Information Sets** to determine not to overwrite an existing connection information set with a set of the same name.

Notes:

1. If the first option is checked, the second option is checked automatically and cannot be unchecked.
2. The Session Data entity will be merged with the existing Session Data entity. When there is a conflict between the imported to the existing Session Data entity, your selection in this checkbox will determine how the Session Data entity will be.
3. When importing an entity to an application that has the same entity name, the existing entity will be overwritten by the new one. The timestamps of the new entity (for example creation and modification date) also overwrite the values of the original entity. These changes take effect immediately, not after reloading the application.

3 Click **Finish**. The entities are imported into the application.

Note: Clicking Cancel on the Progress Bar dialog box will cause the process to be stopped and the entities will not be imported.

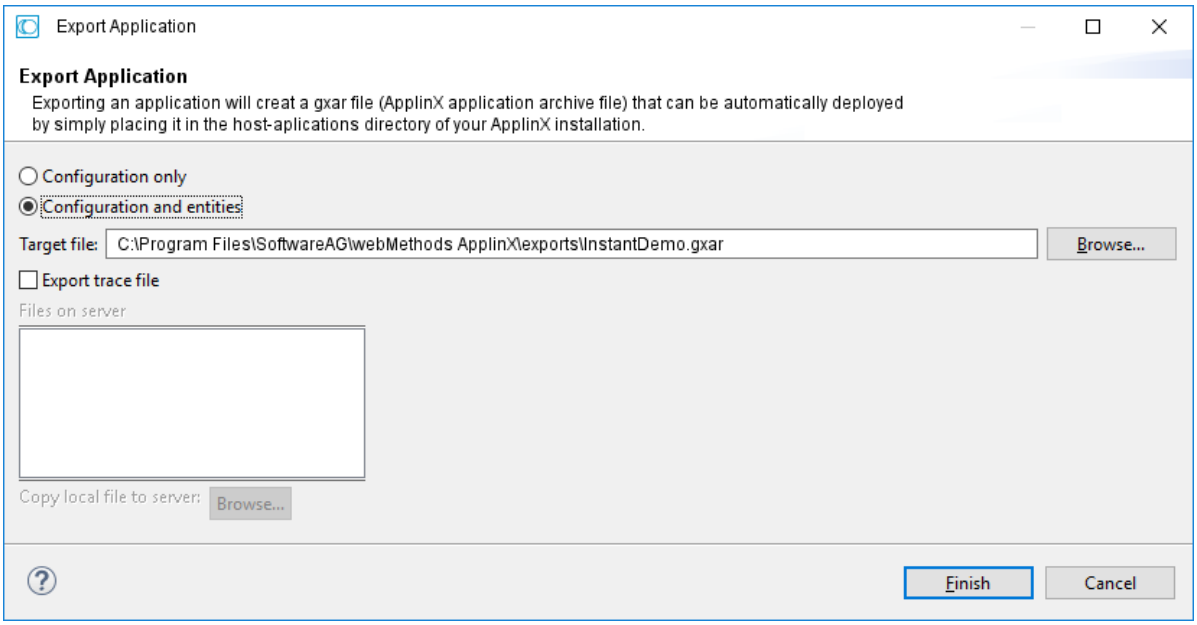
Exporting an Application's Configuration or Entities

The Export Wizard allows you to export an application configuration and its repository residing on various database systems into a standard zip file. This wizard guides you through the steps of exporting data.

Note: When exporting entities, the Session Data entity will always be exported.

To export an application configuration with/without its entities

1 Right-click on the application's node and choose **Export Application**. The Export Application wizard is displayed.



2 Select whether to export just the application configuration or the application configuration together with the entities.

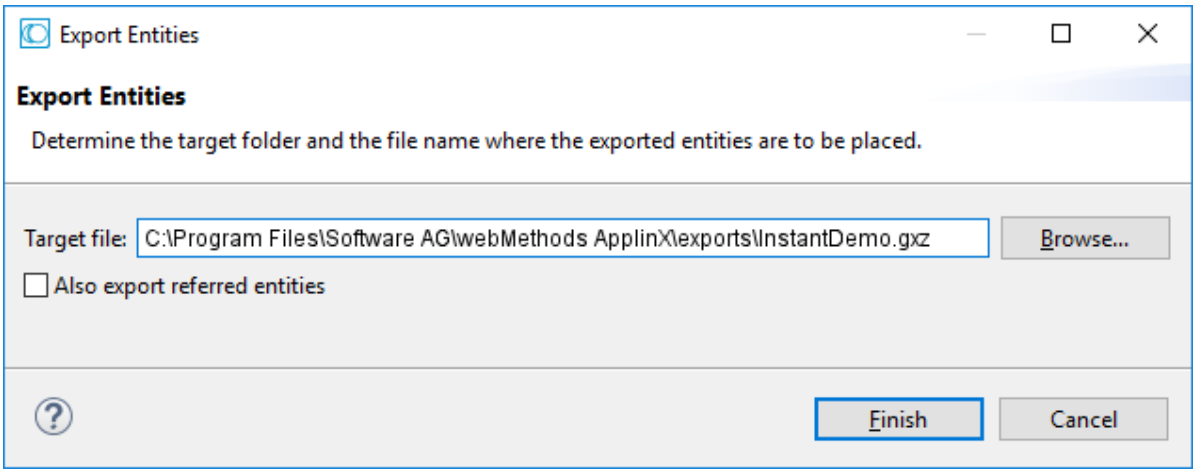
3 Enter a folder name, or browse and select the target folder.

4 Check the **Export trace file** check box to export a trace file. A list of replay (gct) files that are on the server are displayed. Select a file or click **Browse...** to select a local file to copy to the server.

5 Click **Finish**. The application will be exported to the defined folder.

To export entities only

1 Right-click on the application's Repository node and choose **Export Entities**. The Export Entities wizard is displayed.



2 Browse and select the target folder and file name where the exported entities are to be placed.

3 Click **Finish**. The export process will commence and the entities will be exported to the selected folder.

Note: The Session Data entity will always be exported, regardless of the entities selected to export.

Comparing Applications

- Introduction
- Steps

Introduction

With the function **Compare Application...** you can compare two ApplinX applications, typically two versions of the same application. In terms of application lifecycle management, being able to compare two different versions means you can perform impact analysis before deploying a new application version. For example, before deploying to your production environment you can compare your current test application with the production version to see what changes were made, make sure the new application version is complete and that all the included changes were intentional.

Any application that is accessible from the Designer can be compared: loaded or unloaded, remote or local. Output from the compare operation is written to the Eclipse Console view and contains the following information:

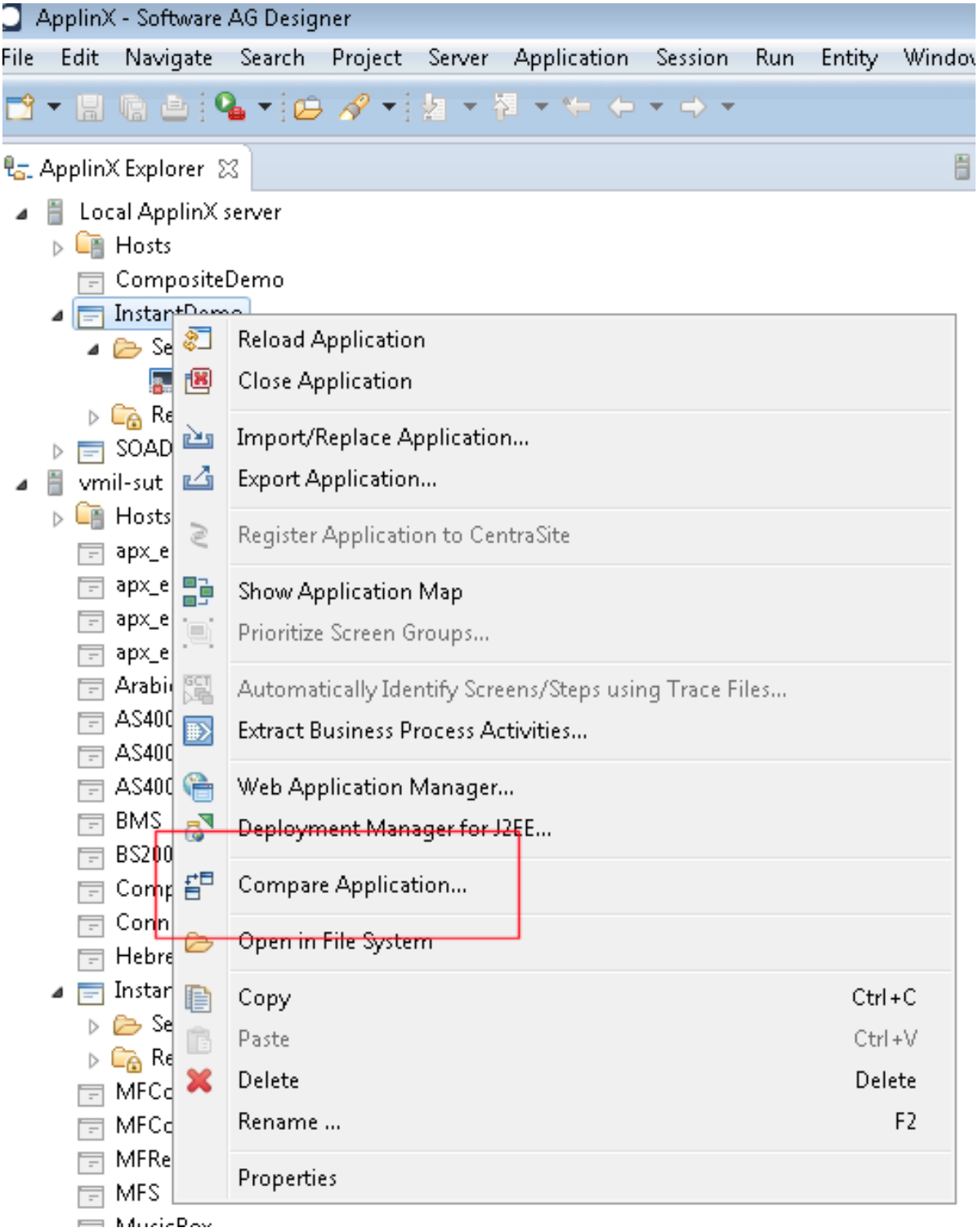
- when the two applications were created/last modified
- entities that exist in one application but not the other (indicated by "=>" and "<=")
- entities that exist in both applications but with differences (indicated by "!=")
- how many entities match (no differences)
- entities that moved location between the two applications
- entities that were renamed between the two applications (entities are identical in content but with different name)

See **Sample Output**.

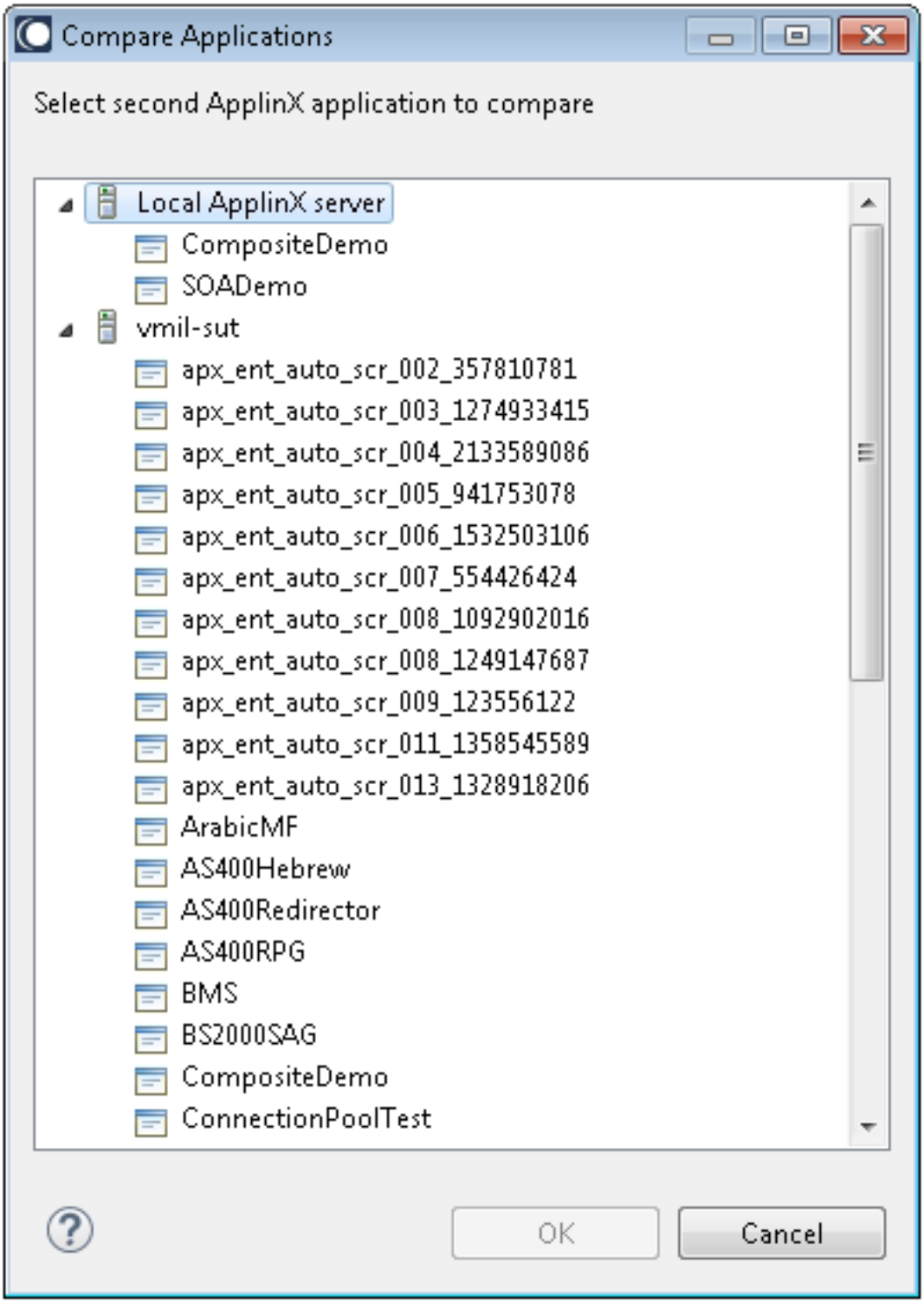
Steps

To compare two ApplinX applications

1 In the ApplinX Explorer, select the first application you want to compare, and from the context menu choose **Compare Application...**.



2 Select the second application you want to compare. The list provided includes all applications accessible from the Designer. These can be loaded or not loaded, and either local or remote.



3 Confirm with **OK** to write the output to the Eclipse Console view. See sample output below.

Sample Output

```
Starting compare process...
Applications loaded successfully


InstantDemo (on Local ApplinX server) and InstantDemo (on APX_Prod) comparison ↩
result:


Summary:
--------
InstantDemo (on Local ApplinX server) Last created: 12/08/2015 09:55:31
InstantDemo (on Local ApplinX server) Last modified: 12/08/2015 09:55:41 By: ↩
Administrator
InstantDemo (on APX_Prod) Last created: 12/08/2015 09:49:16
InstantDemo (on APX_Prod) Last modified: 12/08/2015 09:49:31 By: Administrator


2 entities exist in InstantDemo (on Local ApplinX server) but not in InstantDemo ↩
(on APX_Prod)
2 entities exist in InstantDemo (on APX_Prod) but not in InstantDemo (on Local ↩
ApplinX server)
10 entities are different
81 entities match
3 entities moved their location between compared applications
1 entities renamed between compared applications


InstantDemo (on Local ApplinX server)               InstantDemo (on APX_Prod)    ↩

------------------                                  ------------------           ↩

                                                =>  newScreen (Screen)           ↩

                                               =>  HideTextTODel (Transformation)


customers (Folder)                              <=                               ↩

newTransformation (Transformation)              <=                               ↩



/customers/BrowseCustomers (Screen)             !=  BrowseCustomers              ↩

BrowseCustomersAddress (Screen)                 !=  BrowseCustomersAddress       ↩

BrowseProposals (Screen)                        !=  BrowseProposals              ↩

/customers/CustomerDetail1 (Screen)             !=  CustomerDetail1              ↩
```

```
InsuranceMenu (Screen)                          !=   InsuranceMenu                ↩
MyLogin (Screen)                                !=   Login                        ↩
NaturalMainMenu (Screen)                        !=   NaturalMainMenu              ↩
ProposalDetail1 (Screen)                        !=   ProposalDetail1              ↩
AllGroup (Screen Group)                         !=   AllGroup                     ↩
BrowseCustomersAddress (Screen based table)     !=   BrowseCustomersAddress       ↩


Entities moved:                                                                   ↩
InstantDemo (on Local ApplinX server)                InstantDemo (on APX_Prod)    ↩
------------------                                   ------------------           ↩
/customers/BrowseCustomers (Screen)                  BrowseCustomers              ↩
/customers/CustomerDetail1 (Screen)                  CustomerDetail1              ↩
/customers/BrowseCustomers (Screen based table)      BrowseCustomers              ↩


Entities renamed:                                                                 ↩
InstantDemo (on Local ApplinX server)                InstantDemo (on APX_Prod)    ↩
------------------                                   ------------------           ↩
MyLogin (Screen)                                     Login                        ↩
```

Searching Applications in the Software AG Designer

In addition to searching for entity names and referring entities, you can search within an individual entity, for example for screens and screen groups, path and flow procedures.

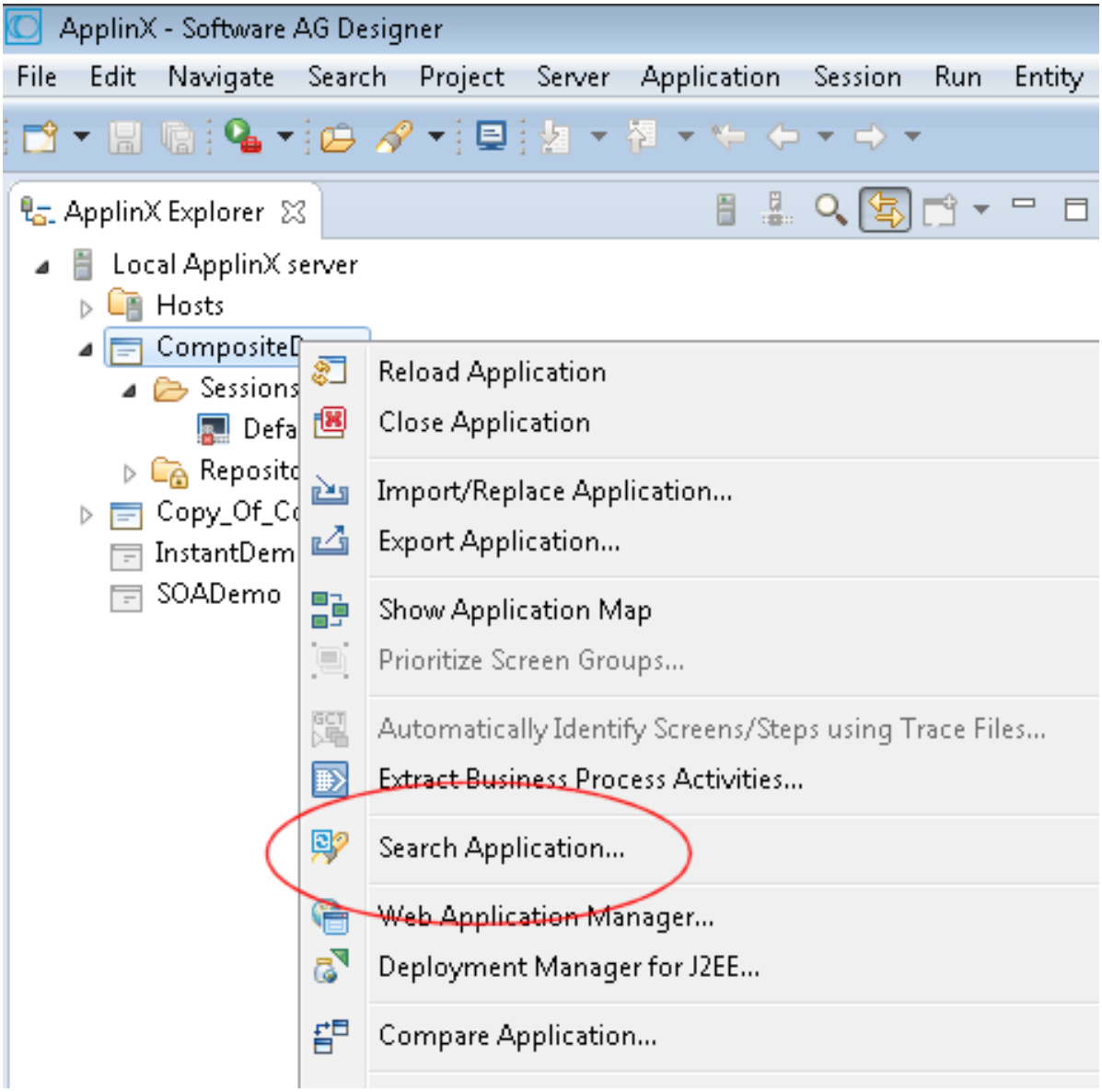
- Starting your Search
- Wildcards

- Filtering

Starting your Search

To start the Designer search

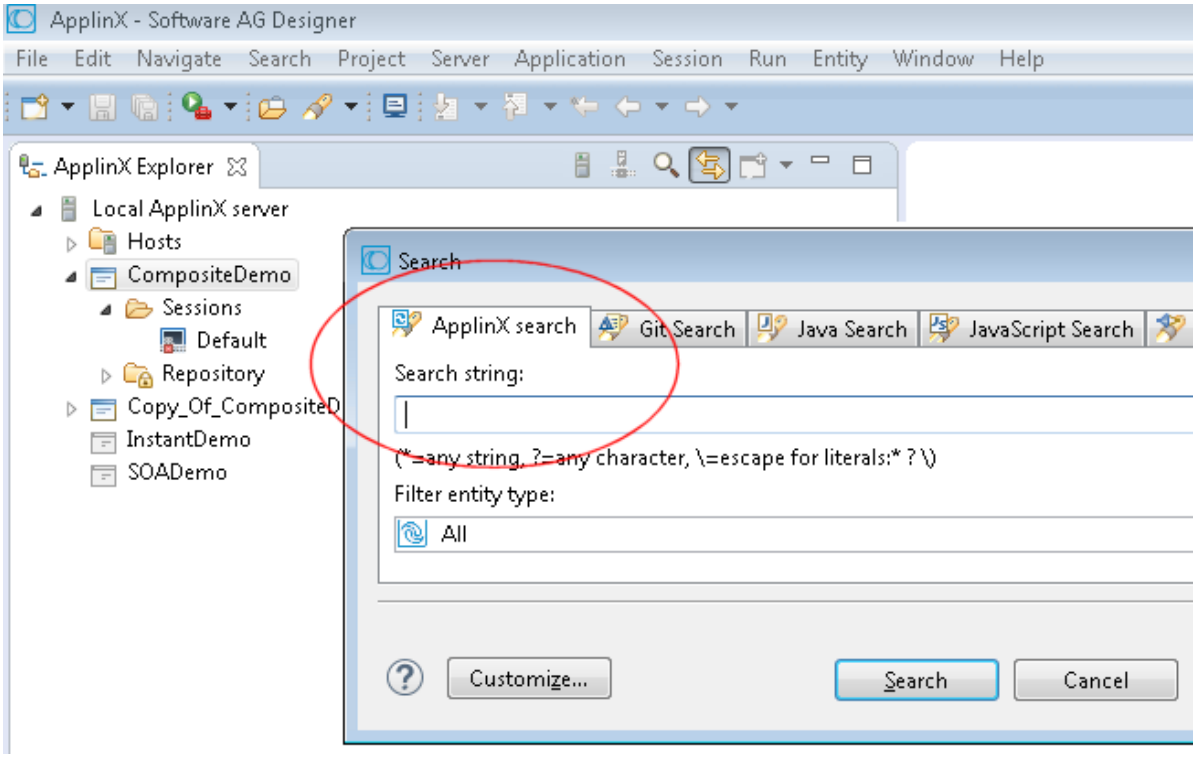
1 In the ApplinX Explorer, select the application you want to search, and from the context menu choose **Search Application...**.



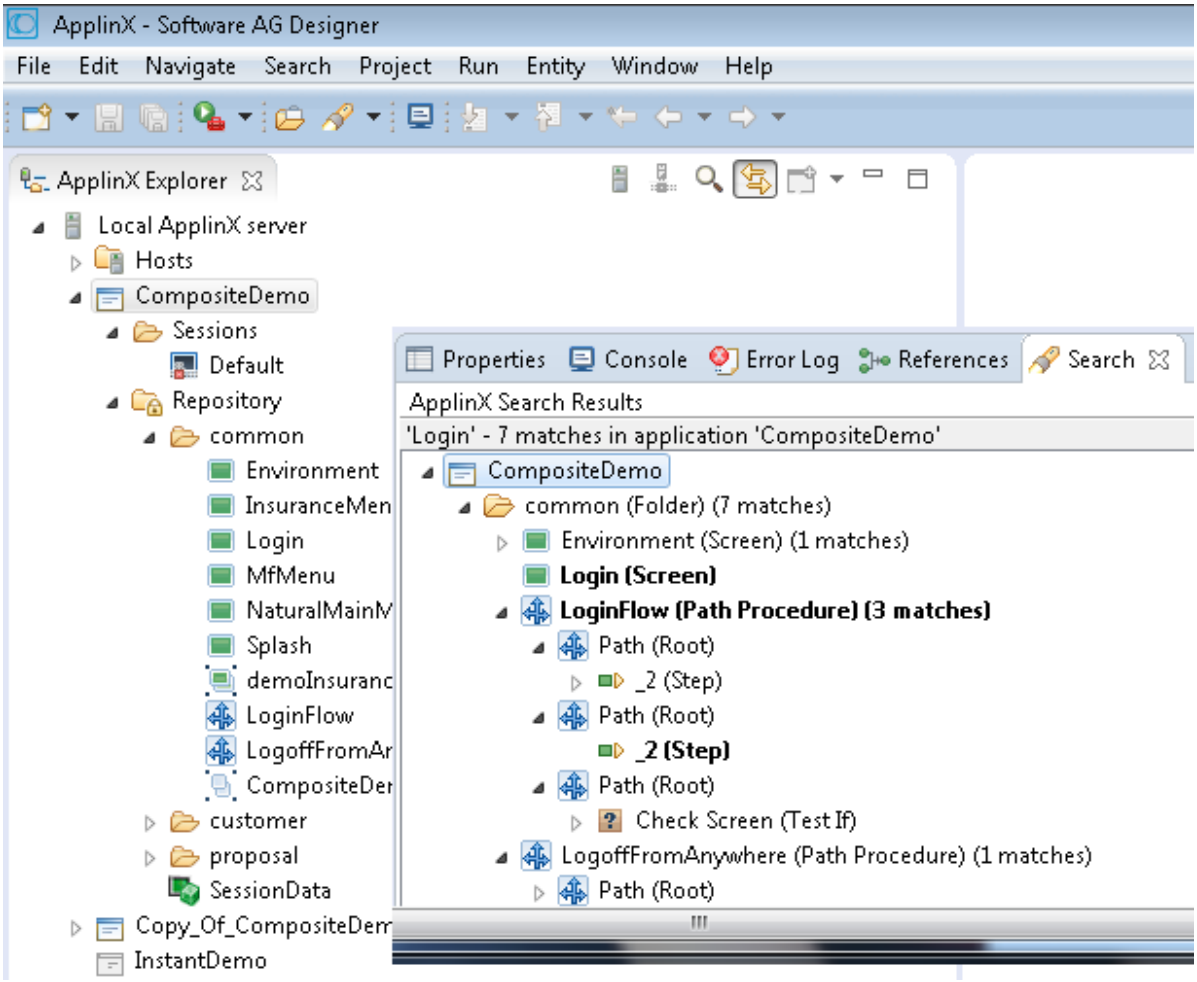
Or:

From the **File** menu, choose **Search** > **ApplinX...**.

2 Open the **ApplinX Search** tab and enter your search string. You can **filter** the search or use **wildcards**.



3 Results are shown in a view. A maximum of 100 hits is returned. You are prompted to refine your search if this number is exceeded.



Wildcards

The following wildcard characters are supported to narrow your search:

*	Asterisk.	Any number of characters. String "log*" returns "log", "logon", "logoff"...
?	Question mark.	Single character. String "b?t" returns "bit", "bot"...
\	Backslash.	Escape character. Must be used if you want to find any of these wildcard characters (*?/) in your string. Example: Enter "my\\string" to search for "my\string".

Note: By default the search behaves as if the string is prefixed and suffixed by an asterisk (any number of characters). Example: String "log" will return "log", "autologOn", "logOff"...

Filtering

You can filter your search by the following entity types:

- Path procedure
- Flow procedure
- Session data
- Data structure
- Screen
- Screen group

Filtering makes sense if your search returns more than 100 results.

Recording Trace Files

The Trace File feature enables recording a file, which traces the connection communication (Web/SOA ApplinX user or Terminal Emulation Proxy) between the ApplinX server and the host, for each connection. It is possible to define whether a single trace file will be created, replacing the previously saved file or whether the data will be saved to a new file for every new user session. Identifying the separately saved files is possible by inserting identifying parameters in the file name (the session ID, creation time and/or connection ID). It is highly recommended to create a separate file for each session/connection or creation time. Note that trace files can be created from within the session definition overriding the application definition. This is recommended as it prevents conflict with other existing sessions.

The trace files can be also defined to be created in folders per Day Month and Year. This is recommended as it prevents conflict with other existing sessions.

Note: Log and trace files can contain sensitive personal data (for example user ID, IP address, etc.). We recommend you check the different trace opportunities provided by ApplinX and delete log and trace files if they are no longer needed. ApplinX will not delete these files automatically; this is your responsibility as user. Use the appropriate tools of the respective operating system.

The trace files can be compressed to save space on the disc. This is particularly suitable when tracing a large number of files on a regular basis.

When required, you can select to encrypt the trace files.

To record a trace connection file

1 In the ApplinX Explorer, right-click on the relevant application and choose **Properties**.

2 Click on the **Host>Recording** node.

3 Check **Record terminal sessions (trace files)**.

4 Check the **Compress (create files in zip format)** check box to compress the file. Compressed files will have the suffix .zip.

5 Check the **Encrypt (using server private key)** check box to encypt the file. In order to encrypt files, you must first define the encryption key (In the Server properties, General tab). Encrypted files will have the suffix .gctx. A file which is both compressed and encrypted will have the suffix .gctx.zip.

6 Choose **Suppress hidden fields** to conceal passwords and hidden fields.

Note: The "Suppress hidden fields" option is not supported when recording using Terminal Emulation Proxy.

7 Provide a name for the file.

8 Check the relevant check box to determine that a new file will be created for each session/creation time or connection ID. It is possible to add the following parameters to the file name instead of using the check boxes: %u will insert the session ID. %t will insert the creation time stamp of the connection. %c will insert the connection ID.

9 Browse and select the location of the folder where the files will be stored. Determine whether sub folders will be created for each year/month/day.

10 Click **OK** to save your changes.

For further details regarding the application properties refer to the Application Reference section.

Working with Offline Sessions

In order to develop, debug and reconstruct specific scenarios, it is possible to replay (GCT) files that have traced the emulation protocol for each user. See **Recording Trace Files**. In order to replay such files, the application configuration definitions must indicate that instead of working online with the host, ApplinX must, when connecting, access and replay the specific file. Note that replay file can be defined from within the session definition overriding the application definition. This is recommended as it does not conflict with activities of other users.

Note: Offline sessions support replaying encrypted and/or compressed files. Refer to **Recording Trace Files** for additional details

To define working with offline replay files

1 In the ApplinX Explorer, right-click on the relevant application and choose **Properties**.

2 Expand the **Host** node and select the **Offline** node.

3 Check the **Work offline** check box.

4 Either enter or browse to select the file name including the full path.

5 ApplinX enables a session replayed by a GCT to simulate the host's communication delay or to predefine a time delay to wait before showing the information (this is because generally, the application is faster when replayed). This is determined in the **Simulate host delay** field. Available values: No delay, Simulate host, 500- 10000 ms (by default No delay is selected).

6 Click **OK** to save your changes.

For further details regarding the application properties refer to the Application Reference section.

Changing the Initialization Mode of an Application

To change the initialization mode of an application

1 In the ApplinX Explorer, right-click on the relevant application and choose **Properties**.

2 Ensure that the **General** node is selected.

3 In the **Initialization mode** field, select the relevant initialization mode:

When first accessed
Loaded when the code that initializes startup is called.

Automatic
Loaded when the server is started.

4 Click **OK** to save your changes.

For further details regarding the application properties refer to the Application Reference section.

Setting Session Time-Out for Idle Users

To set the time-out for idle users

1 In the ApplinX Explorer, right-click on the relevant application and choose **Properties**.

2 Select the **Host** node.

3 Select whether the session time-out will be unlimited or disconnected after a specific number of seconds.

4 Click **OK** to save your changes.

For further details regarding the application properties refer to the Application Reference section.

Handling Flickering of Host Screens

It is necessary to use the Flickering of Host Sessions feature when one of the following happens:

- In the browser, a blank screen is displayed when navigating between two host screens.

  Note: Refer to Blank Screen Timeout in General Host parameters for additional information on handling blank screens.

- In the browser you are required to submit the [ENTER] key (or any other key) twice in order to navigate to the next host screen.

The initial need for Flicker arises when specific host screens are received 'split' between several buffers of data. Thus ApplinX Server needs to be informed to wait an additional amount of time for the complete screen to arrive. This additional amount of time is defined (in milliseconds) in the Flicker parameter in the Host node of the Application Properties dialog box. The flicker setting applies to the entire ApplinX application, meaning that if the flicker is set to 500ms, after each host transaction the flicker time will be added to the communication time. In other words, the entire application will be 'slowed down' by the flicker time. Therefore this value should only be set for the entire application according to the following guidelines:

As a rule of thumb, there is no reason to use the flicker setting in the application configuration for block mode hosts. Specific 'problematic' screens should be handled using wait conditions, both in navigation paths and in Web applications. For less specific cases (when a wait condition for a specific screen cannot be used), it is possible to set the flicker setting only for certain actions (using either the path dialog or the Base Object API in Web applications).

To define the flicker time

1. Click the relevant application in the ApplinX Explorer.
2. Right-click an application and choose Open. The Application dialog box is displayed.
3. In the Host tab enter a value in the Flicker field. Possible values are from 0 to 10000 ms.
4. Click Apply to save all changes, without closing the dialog box. Click OK to save changes and close the dialog box. You will be required to confirm the change.

For further details regarding the application properties refer to the Application Reference section.

Defining ApplinX RPC Application Parameters

The RPC connections pool is used for holding pre-connected RPC connections. When an RPC connection is required to execute a program, a connection from the pool is used, saving the connection time and executing the program immediately. This feature is available for AS/400 hosts only. This feature is available in SOA applications only.

To set the RPC pool connections settings

1 Open the *Application Properties* dialog box and click on the **Host>RPC** node.

2 Choose **Use Connections Pool**.

3 Configure the RPC connection parameters. Refer to RPC Connection Parameters for further details regarding the fields in this dialog box.

4 When required, configure the username and password needed to connect to the AS/400 programs. Refer to Host Configuration Parameters: RPC.

Defining Host Keys

This feature (relevant only for Web enabling applications) allows you to define a common pattern for the host keys appearing in host screens (it is also possible to define more than one pattern, in cases where the host uses different formats for displaying screen keys). The defined pattern is used by ApplinX Server to automatically analyze the keys present in each screen, and display them in the Session Viewer (as buttons) or in Web application pages (both instant and generated pages), as buttons or hyperlinks. The developer can further customize the appearance of the keys and the way to handle the screen rows containing the original keys.

- Using an Existing Host Key Pattern
- Using Customizable Host Key Pattern TWO LINES
- Defining a Custom Host Key Pattern
- Customization of Host Keys
- Host Keys Viewed in an Instant Web Page
- Host Keys Viewed in a Generated Web Page

Note: For Natural-UNIX hosts, there is no need to define a pattern for PF keys; the definitions are provided inside the protocol, so there is no need for an “external” configuration.

Using an Existing Host Key Pattern

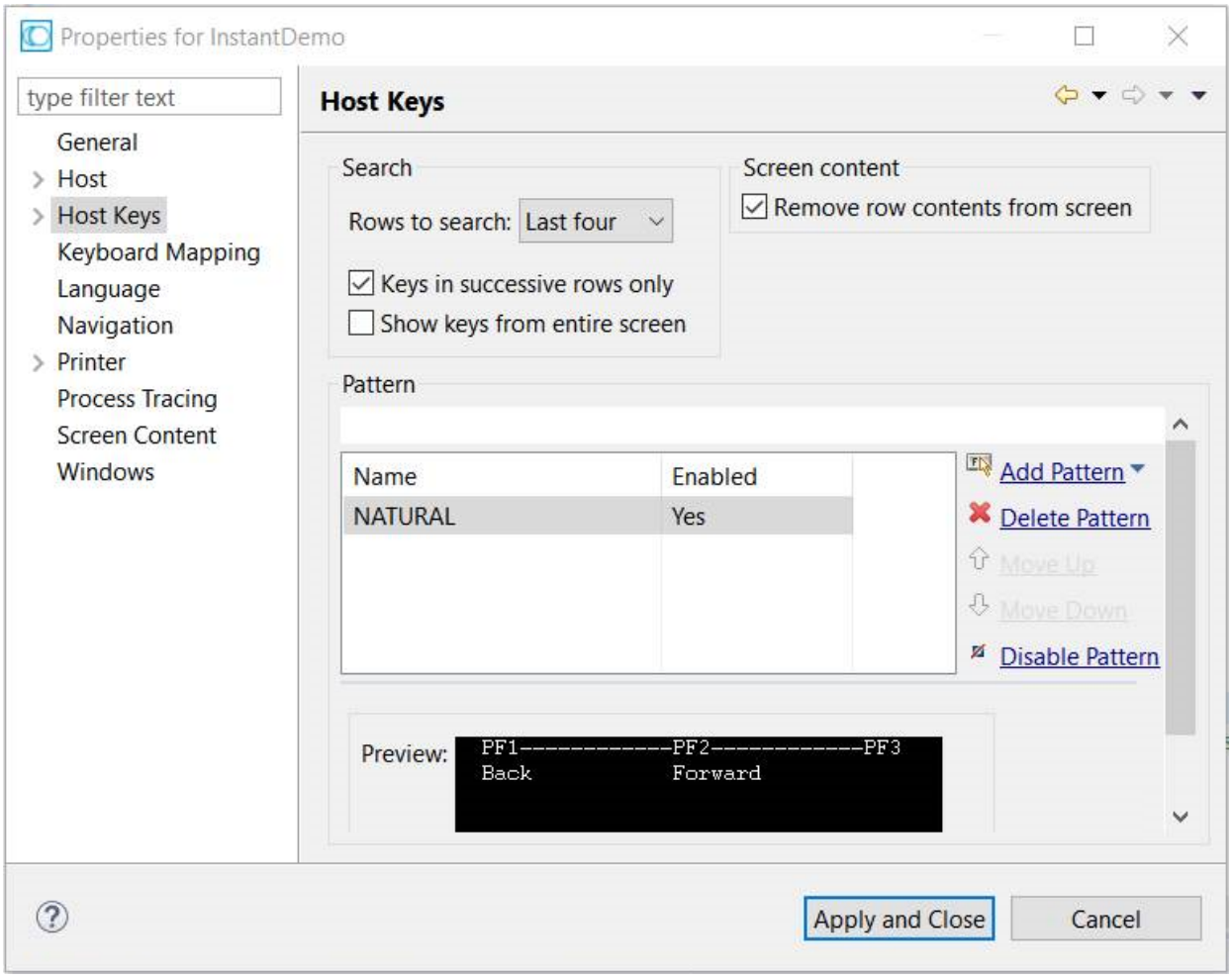
Use this option when the pattern of host keys in the host application matches one of the existing patterns, as is usually the case.

To use an existing host key pattern

1 Open the Application Configuration dialog box (see **Editing an Application's Configuration**).

2 Click the Host Keys tab.

3 Click on the arrow next to **Add Pattern** to select a pattern from the list of exiting patterns and see its preview. The following screen is an example of the NATURAL host key pattern.



Note: Adding the Natural pattern for PF keys will also identify the PA keys pattern, if applicable.

All fields are described under *Host Keys* in the Reference Guide.

Using Customizable Host Key Pattern TWO LINES

From ApplinX version 10.7, a pre-defined host key pattern **TWO LINES** is available. Additional fields are provided to customize the pattern.

To use pre-defined and customizable host key TWO LINES

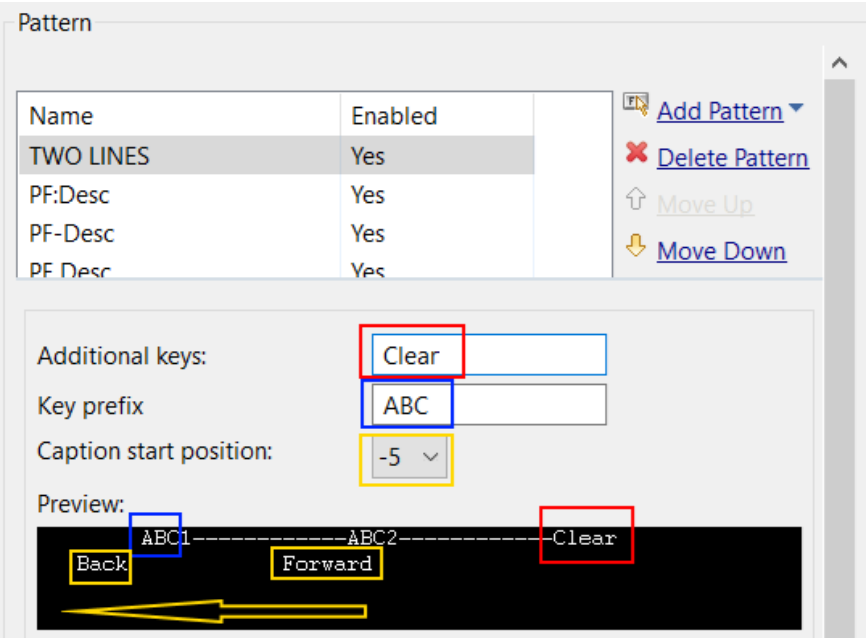
1 Open the **Host Keys** screen as described above.

2 Choose pattern **TWO LINES**.

3 Enter values for

- **Additional keys**
  One or more host keys not suffixed with a number. Multiple values are separated by a comma ','.
- **Key prefix**
  Prefix string for the host key. You can add more keys, separated by a comma ','.
- **Caption start position**
  Starting position of the caption, relative to the host key. Valid values in range -5 to 0.

The result is shown in the **Preview**:



All fields are described under *Host Keys* in the Reference Guide.

Defining a Custom Host Key Pattern

Use this option only if the pattern of host keys in the host application does not match any of the predefined patterns.

To define a custom host key pattern

1 Open the Application Configuration dialog box (see **Editing an Application's Configuration**).

2 Click the **Host Keys** tab.

3 Click **Add Pattern** to add a new pattern.

4 Assign a name for the new pattern in the pattern column.

5 Define the **Key**, **Spaces**, **Internal separator** and **Key separator**. As you define these parameters, the preview is updated and displayed in the **Preview** pane.

All fields are described under *Host Keys* in the Reference Guide.

Customization of Host Keys

This feature allows you to define:

- Host keys that are not available using the standard PC keyboard, such as [help], [clear] or [reset].
- Keys that need to be submitted when the cursor is positioned in a specific position on the host screen.
- Combinations of keys and text to be sent to the host simultaneously.

Note: In standard terminal emulations it is also possible to define keyboard mappings of host keys to PC keyboard keys. To implement the same functionality in ApplinX, define a button that sends the mapped host key. For example, if in the terminal emulation, [clear] was mapped to the keyboard combination CTRL-ESC, in ApplinX define a button that sends [clear].

To add customized Host Keys

1 Right-click on the relevant application, and choose Properties.

2 Click the Host Keys, Manual tab.

3 Click on the Add Host Key hyperlink.

4 Enter a caption for the host key.

5 Enter the host key in square brackets or select a host key from the list of standard host keys.

6 Select the position of the cursor on the screen once the host key has been sent.

Host Keys Viewed in an Instant Web Page

By default, defined patterns will be searched, PF lines will be omitted and keys will be added as buttons or hyperlinks. Refer to Customizing the Host Keys.

Host Keys Viewed in a Generated Web Page

By default, defined patterns will be searched, PF lines will be omitted and keys will be added as buttons or hyperlinks. These options can be controlled in the Generation dialog box, Host Key control.

Mapping Keyboard Keys

It is possible to map specific keyboard keys to the Keyboard Mapping tab in the Application Properties dialog box.

To define Keyboard Mappings

1 Right-click on the relevant application and choose **Properties**.

2 Focus on the Keyboard Mapping tab.

3 To map one of the existing mappings, locate it in the list of keyboard mappings and ensure that it is marked as Enabled. If it is not enabled, click on the Enable Key hyperlink.

4 To add a new keyboard mapping, click on the Add Keyboard Mapping hyperlink.

5 Enter the key combination in the Keyboard field.

6 Either select the host key from the standard list of host keys or enter a host key in the Host key field. Ensure you place square brackets around the key.

Note: When using the keyboard keys within the web application, the CTRL+N and CTRL+K keys are blocked by default as they cause multiple browser windows to use the same session (this can be manually set in the config/gx_keyboardMappings.xml file).

7 Relevant for VThosts only: Define a sequence of hexadecimal numbers to relevant keys.

Key combinations defined here may conflict with Eclipse key bindings. When such a conflict occurs, the key combination defined here will be ignored during runtime. To see the list of Eclipse key bindings, open the **Windows>Preferences** menu, expand the **General** tab and choose **Keys**. A key binding which appears in this list, and whose `When` field is set to "In Windows" or "In Dialogs and Windows" will cause a conflict.

When such a conflict occurs, either redefine the keyboard mapping in ApplinX, or change the Eclipse key binding. The Eclipse key binding can be changed either by choosing a different option in the `When` field (the key binding will be disabled and it will not be possible to enable it), or by editing the *plugin.xml* file of the ApplinX plug-in (add the new key binding in the org.eclipse.ui.bindings extension point with ApplinX scheme ID and empty commandId. This way, whenever the ApplinX key binding scheme is activated, the problematic key bindings will be disabled, but in other schemes the key bindings will be enabled.)

Supporting RTL Languages

The **Application Configuration>Language** node enables you to define the language used in the application as well as direction settings, relevant mainly for right-to-left languages.

To configure RTL language settings

1 In the ApplinX Explorer, right-click on the relevant application and choose **Properties**.

2 Select the **Language** node.

3 Select the application's language.

Note: The following settings are relevant for right-to-left languages. The option you select is the default setting for all the screens, but can be changed for a specific screen in the relevant screen, in the Screen Direction tab.

4 Set the **Screen direction**. Relevant for mainframe hosts only. The screen direction of right-to-left languages differs according to the original host settings, and when incorrectly set, can cause the screen to be illegible. In order to correct this, define the suitable screen direction.

5 Set the **Typing direction**. Right-to-left languages may display typed-in text in the Display Session View and HTML fields, aligned to the left of the field. In order to display the text aligned to the right, select the right-to-left option.

6 Set the **Tab direction**. When pressing the TAB button the cursor moves to the next consecutive field. The direction the cursor moves (moving to the next field to the left or moving to the next field to the right) must be correctly defined in order to preserve the screen logic.

Recording Steps while Navigating

The Application Map view enables viewing a navigation map between screens. Once the navigation option is selected, in the Application Properties, Navigation tab, the application map will record the navigation steps between the screens. See also Navigation Parameters and Application Map.

Defining Printer Parameters

Refer to Working with the Printer Session.

Creating Logs for Tracing Application Processes

The Application Process Tracing is used to provide a log of performance times of processes such as procedures, paths and programs. This can give a more specific indication (pinpoint the exact process) as to the cause of application performance problems. The application process tracing information can be saved as a txt and/or csv file. These files are located in the ApplinX>log directory.

Note: The Host times that the tracing application process displays, are only relevant for hosts that are not character mode hosts.

Configuring the Tracing Application Processes Feature

To enable the tracing feature

1 Open the *Application Properties* dialog box and click on the **Process Tracing** node.

2 Check the **Enable process tracing** check box.

3 **Basic logging** is selected by default, and logs data at the entity level (when a procedure, path or program started and when they were completed). To log more detailed data, at the step or node level, select **In-depth logging**.

4 Select **Log each process from beginning to the end** to log the entire process. Select **Log the process summary only** to log a summary of the process.

5 Select **Only log processes completed in more than...** to log only the processes that take longer than the amount of time defined in the milliseconds field.

6 Select the file type: **Simple text file (txt)** or **Comma separated values (csv)**. The csv format is suitable when needing to access the data from an external database or from other applications such as Excel or Access.

7 Provide a name for the file.

8 Check the relevant check box to determine that a new file will be created for each creation time or process ID.

Analyzing the Application Process Tracing Output

Once the application process tracing feature is enabled, trace files (txt and/or csv) are automatically created in ApplinX>log directory.

Note: Host times are only relevant for hosts that are not character mode hosts.

- TXT File
- CSV File

TXT File

The TXT file includes:

- Time and date when the event occurred.
- Thread name.
- Message type: DEBUG, INFO, WARN or ERROR.
- Process ID: A unique ID of the process.
- Node type: HOST-CON, PATH, STEP, FLOW, NODE, EXWEBSVC or PROGRAM.
- Event type: STRT, FIN and ABRT.
- Full node type name: Such as Host connection as oppose to HOST-CON. The full node type is indented according to the depth in the current process. This is determined by the process ID.
- Application: The application name.
- Node name: Full node name, including the relevant folder name.
- Full event type name: Started, finished and aborted.

According to the event type and node type, additional data may be displayed:

- All events that the event type is finished or aborted will display the total time.
- Host connection events will display the user name.
- Finished or aborted paths will display the total number of steps, total host time and total wait time (wait time is the time that ApplinX waited after the final response from the host, due to wait condition definitions). All these parameters include all inner path parameter details.

- Finished or aborted steps will display total steps, total host time and total wait time.

CSV File

The CSV file includes:

- Time and date when the event occurred.
- Process ID: A unique ID of the process.
- Depth of node
- Type of node: Host connection, Path, Step, Flow, Node, External Web Service or Program.
- Application: The application name.
- Node name: Full node name, including the relevant folder name.
- Event type: Started, finished and aborted.
- User name, Total time, Host time, Wait time and Number of steps in the completed path will be displayed (in the above order) in the following cases:
  - All events that the event type is finished or aborted will display the total time.
  - Host connection events will display the user name.
  - Finished or aborted paths will display the total number of steps, total host time and total wait time (wait time is the time that ApplinX waited after the final response from the host, due to wait condition definitions). All these parameters include all inner paths.
  - Finished or aborted steps will display total steps, total host time and total wait time.

Repository

The ApplinX repository is an internal database used to hold ApplinX entities (metadata) such as the Screens, Paths, Connection Pools, Programs and Procedures.

As an ApplinX user, you may like to divide applications into folders within your repository, according to the specified interests and needs, such as development teams, sub-applications etc. The same folder contains different types of entities and sub-folders varying according to each application. This allows you greater flexibility organizing your entities.

The repository can be read only. It is recommended to use a read-only repository for production.

Selecting a Read-only Repository

To set up a read-only repository

1 Open the application.

2 Right-click on the repository node.

3 Choose **Lock Repository (read only)**

Defining Host Windows

The Windows definitions are used to correctly identify screens and to open host windows as separate pop-up windows. In the Application Properties it is possible to define the Host Windows per the application.

When adding a window you are required to select the type of modal windows the host sends. There are two types: Reversed Video or Frame. For applications that have more than one level of windows (a window within a window), where each level is defined with a different Window Type, be sure to define the windows in the correct order.

Note: When the host is a Natural UNIX host, this tab is disabled, as the windows' definitions are included in the Natural-UNIX protocol and do not require being defined via ApplinX.

To define windows

1 Right-click on the relevant application and choose Properties.

2 Add a Frame or Reversed Video

3 Configure the parameters as detailed in Application Configuration Parameters, Windows tab

3 Using the ApplinX REST API

Using the ApplinX REST API you can create and disconnect a session, get a screen and update a screen. Four authentication modes are provided to call the API: ApplinX, LDAP, OpenID Connect and Host Authentication.

Getting Started

You can get the API description (based on the OpenAPI Specification) of the REST API by sending a `GET` request to the following URL:

```
http://{host}:{port}/applinx/rest/swagger.json
```

Example:

```
http://localhost:2380/applinx/rest/swagger.json
```

We recommended you paste the response output in the Swagger Editor in the following URL:

```
https://editor.swagger.io/
```

This gives you all the information about the operations and models of the API, as well as generating an out-of-the-box new client (in your favorite language) that consumes the API.

Configuration

The ApplinX REST API contains a configuration file *rest-api-config.json* in folder *WEB-INF\config*. Use this file to define the configuration parameters:

Parameter	Description
`serverURL`	ApplinX server URL.
`applications`	Names of applications that the sessions can be connected to.
`security`	
`auth`	Four authentication modes are supported. These are described in more detail below. ■ ApplinX Authentication (using the ApplinX Administator) ■ LDAP ■ OpenID Connect ■ Host Authentication, for hosts such as Natural-UNIX that already have an authentication step

Parameter	Description
	Note: You can also disable authentication with setting `auth=disabled`.
`groups`	Defines groups for different user types.
`admins_group`	
`developers_group`	
`users_group`	

Session Management

Creating a Session

To create a session (connection)

- Send a `POST` request to the following URL:

```
http://{host}:{port}/{basePath}/rest/session
```

For example:

```
http://localhost:2380/applinx/rest/session
```

Request Body:

```
{

"applicationName": "string", (name of application that the session will be ↩
connected to)
"connectionPool": "string", (Optional)
"sessionDescription": "string"(Optional)
 "naturalUsername": "string", (required for Natural hosts).
 "naturalPassword": "string", (required for Natural hosts).
"naturalNewPassword": "string", (fill this if you'd like to change the host ↩
password).
"failOnWarning": boolean, (don't connect session if getting a warning when trying)
 "options": {  (Optional GXBaseObjectConstants)
   "additionalProp1": "string",
   "additionalProp2": "string",
```

```
  },
}
```

Notes:

1. Parameters `naturalUsername` and `naturalPassword` are required only if you configured `hostAuth` as your authentication mode. See **Host Authentication**.
2. See the available BaseObject constants which apply to the options parameter: `GXBaseObjectConstants`

Authorization Header:

If you are using ApplinX's Administrator or LDAP authentication, add `Basic Auth` schema to the Authorization header.
Example:

```
"Basic lfhsdadkankdnaadacxzczca"
```

The second word in the string is you username/password encoded in Base64. For example, in JavaScript use

```
var auth = 'Basic ' + btoa(username + ':' + password); ↩
```

and add the `auth` string to the Authorization header of the request.

Response Body:

```
{
 "token": "string",
 "keyboardMapping": [
   {
     "keyCode": 0,
     "additionalKey": 0,
     "targetFunction": "string"
   }
 ],
 "message": "string",
 "redirectUri": "string"
}
```

Note: `"redirect_uri"` will be returned only if using OpenID Connect authentication. See **OpenID Connect Authentication**.

Important: The response body contains a variable called 'token'. This token represents your session and will be used in all subsequent requests. The token should be added to the authorization header using `bearer token` schema.

Example:

```
var auth = "Bearer <token>"
```

Replace <token> with your token value and add the auth string to the Authorization header of all subsequent requests.

To disconnect a session

- Send a `DELETE` request to the following URL:

```
http://{host}:{port}/applinx/rest/session
```

Getting the Current Screen

When a connection is established, using the create session request, you can get the current screen, as well as update the screen and navigate to the next.

To get the current screen

- Send a `POST` request to the following URL:

```
http://{host}:{port}/applinx/rest/screen
```

Request Body:

`options` are optional key-value pairs, for example:

```
{
  "options": {
    "GX_VAR_TRIM_FIELDS": "none",
    "GX_VAR_TRIM_MODE": "bothSides"
  }
}
```

Note: See **Important Note** on adding the token in the authentication header under *Creating a Session*

Updating the Current Screen

To update the current screen and navigate to the next

- Send `PUT` request to the following URL:

```
http://{host}:{port}/applinx/rest/screen
```

Request body (required):

```
{
  "fields": [
    {
      "name": "string", // name is interchangeable with position (see below).
      "value": "string", (required)
      "index": 0, (Only for multiple field)
     "position": { // if the field has name, position is interchangeable with ↩
name.
        "column": 0,
        "row": 0
      }
    }
  ],
  "sendKey": "string",
  "cursor": {
    "fieldName": "string", (optional, interchangeable with position).
    "position": {
      "column": 0,
      "row": 0
    }
  }
}
```

Example 1:

```
{
  "fields": [
    {
      "value": "Hello World!",
      "position": {
        "column": 15,
        "row": 21
      }
    }
  ],
  "sendKey": "[enter]",
  "cursor": {
    "position": {
      "column": 11,
      "row": 3
```

```
    }
  }
}
```

Example 2:

```
{
  "fields": [
    {
      "name": "FIELD02",
      "value": "Hello world!",
    }
  ],
  "sendKey": "[enter]",
  "cursor": {
    "fieldName": "FIELD03"
  }
}
```

Request header:

Screen ID (required)

The current screen ID obtained in a get screen response.

To get information on whether there is a session connected and the authentication mode defined in the configuration file

- Send a `GET` request to the following URL: *http://{host}:{port}/applinx/rest/info*.

Note: See **Important Note** on adding the token in the authentication header under *Creating a Session*

Web Application Logging

One of the REST API purposes is to be used by a web application. Sometimes, complex web applications need to monitor the front-end application and the web pages on the server-side because web browsers cannot directly log messages to a file on a local machine. The REST API exposes an endpoint that enables web applications to send logs from the browser to file under ApplinX installation folder through an HTTP call. The log file, *javascript_log.txt*, is placed in the installation folder *SoftwareAG/ApplinX/log*.

To log a message

- Send a `POST` request to the following URL:

```
http://localhost:2323/applinx/rest/logger
```

Request body:

```
{
   "timestamp": "string",
   "level": 0,
   "fileName": "string",
   "lineNumber": "string",
   "message": "string"
   "additional": [
     "string"
   ],
}
```

- The `fileName` and `lineNumber` parameters correspond to the JavaScript file after compilation.
- You can send any additional data with the `additionalData` parameter. The additional data will be aggregated and shown in the log output.

Sample output in javascript_log.txt:

```
Date and time of event       level     filename:line   username        session ↩
ID                                                                  message       ↩
  additional data
|----------------------||-----||----------||------------||-----------------------------------|---------||--------------------------|
2021-03-24T09:45:30,883 FATAL ↩
[main.js:3608][administrator][398FD26A58C9774CEBD6EE9A0CC059DF] hello world ↩
data1, data2, data3, data4 ........
```

Authentication

The ApplinX REST API supports the following authentication modes:

- ApplinX Authentication
- LDAP Authentication
- OpenID Connect Authentication
- Host Authentication

- Disabled

ApplinX Authentication

To configure ApplinX authentication

- In the the configuration file *rest-api-config.json* configuration file, set security parameter `auth` to "`applinx`". Example:

```
{
  "serverURL": "applinx://localhost:2323",
  "applicationName": "InstantDemo",
  "security" : {
      "auth" : "applinx",
      "groups" : {
          "admins_group" : "Supervisors",
          "developers_group" : "Developers",
          "users_group" : "Everyone",
      }
  }
}
```

Note: Authentication of type `applinx` uses the users and groups from the ApplinX Administrator tool.

LDAP Authentication

This authentication mode uses a third-party LDAP server.

To configure LDAP authentication

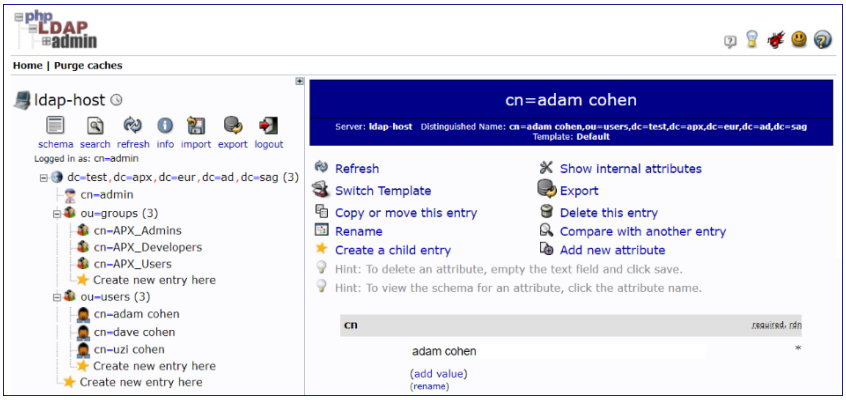
- In the the configuration file *rest-api-config.json* configuration file, set security parameter `auth` to "`ldap`" and provide the LDAP environment variables. Example:

```
{
    "serverURL": "applinx://localhost:2323",
    "applicationName": "InstantDemo",
    "security": {
        "auth": "ldap",
        "groups": {
            "admins_group": "APX_Admins",
            "developers_group": "APX_Developers",
            "users_group": "APX_Users"
        },
        "ldap": {
            "url": "ldap://localhost:123",
            "dn": "cn=admin,dc=test,dc=apx,dc=eur,dc=ad,dc=sag",
```

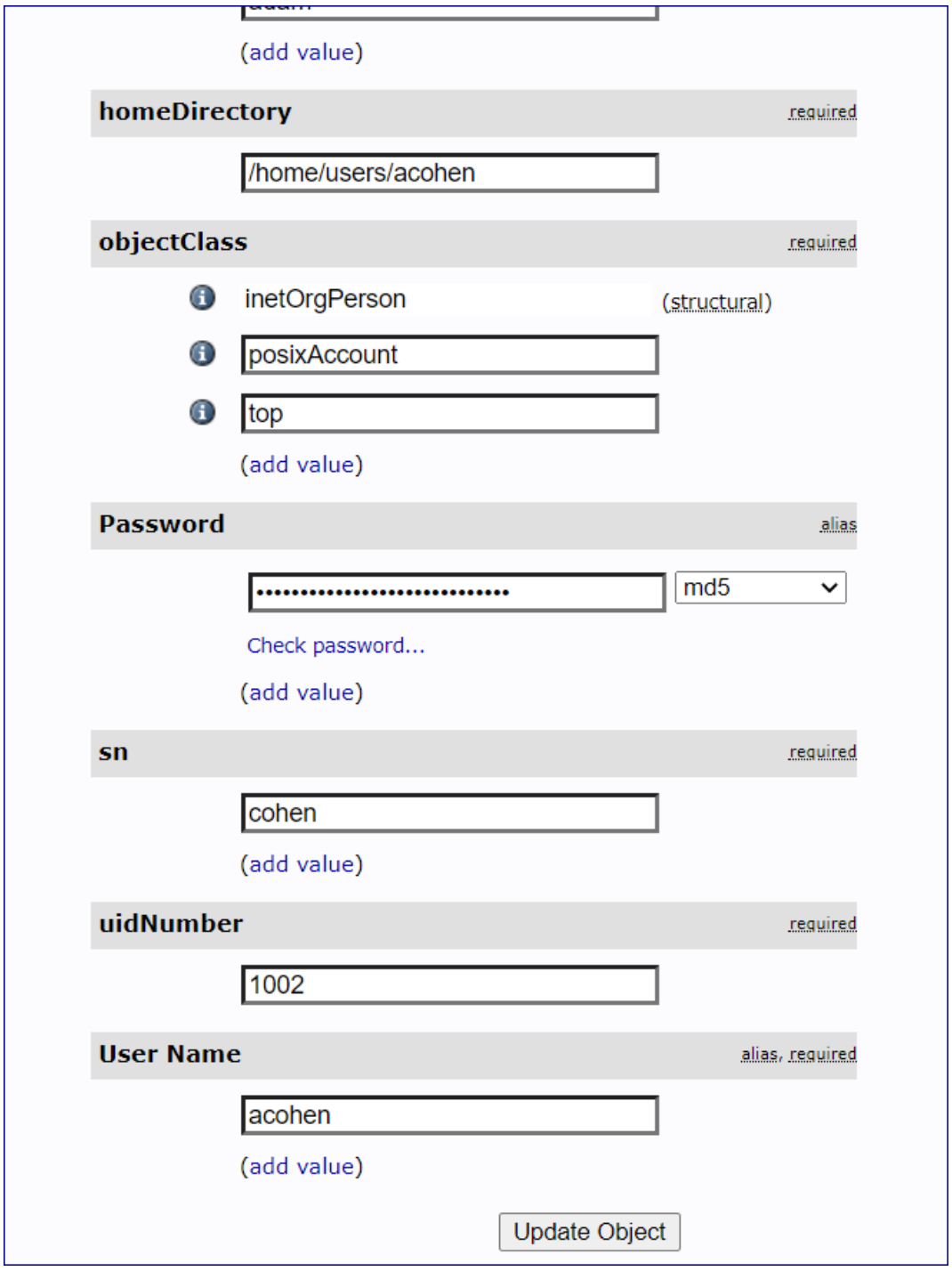
```
            "password": "passwordExample",
            "authentication": "simple",
            "filter_user": "(uid={})",
            "filter_admins_group": "(&(cn=APX_Admins)(memberUid={}))",
            "filter_developers_group": "(&(cn=APX_Developers)(memberUid={}))",
            "filter_users_group": "(&(cn=APX_Users)(memberUid={}))",
            "context_users": "dc=test,dc=apx,dc=eur,dc=ad,dc=sag",
            "context_groups": "dc=test,dc=apx,dc=eur,dc=ad,dc=sag"
        }
    }
}
```

LDAP Environment Variable	Description
`url`	The LDAP server's address.
`dn`	The DN of read-only account, if not anonymous.
`password`	The password of read-only account, if not anonymous.
`authentication`	Either "`simple`" if a read-only search account is required, or "`none`" if anonymous.
`filter_user`	Filter to find user's record; note that {} is replaced by the user-supplied ID.
`filter_admins_group`, `filter_developers_group`, `filter_users_group`	These are filters for our three application roles... You might have to use 'member' instead of 'memberUid'.
`context_users`	Optional path to start the user search somewhere more specific than the root.
`context_groups`	Optional path to start the group search somewhere more specific than the root.

This is the LDAP server we connected to in the above example, showing the context of the groups and users:



After everything has been configured in the configuration file, the end users log in by sending their username (which in this example is the `uid` we configured inside the `filter_user` parameter) and password.



In this example we have a user with **User Name** `"acohen"`. To log this user in, send `acohen:pass`. Send the `username:password` in the Authorization Header of the create session request.

OpenID Connect Authentication

Configuring OpenID Connect Authentication

To configure OpenID Connect authentication

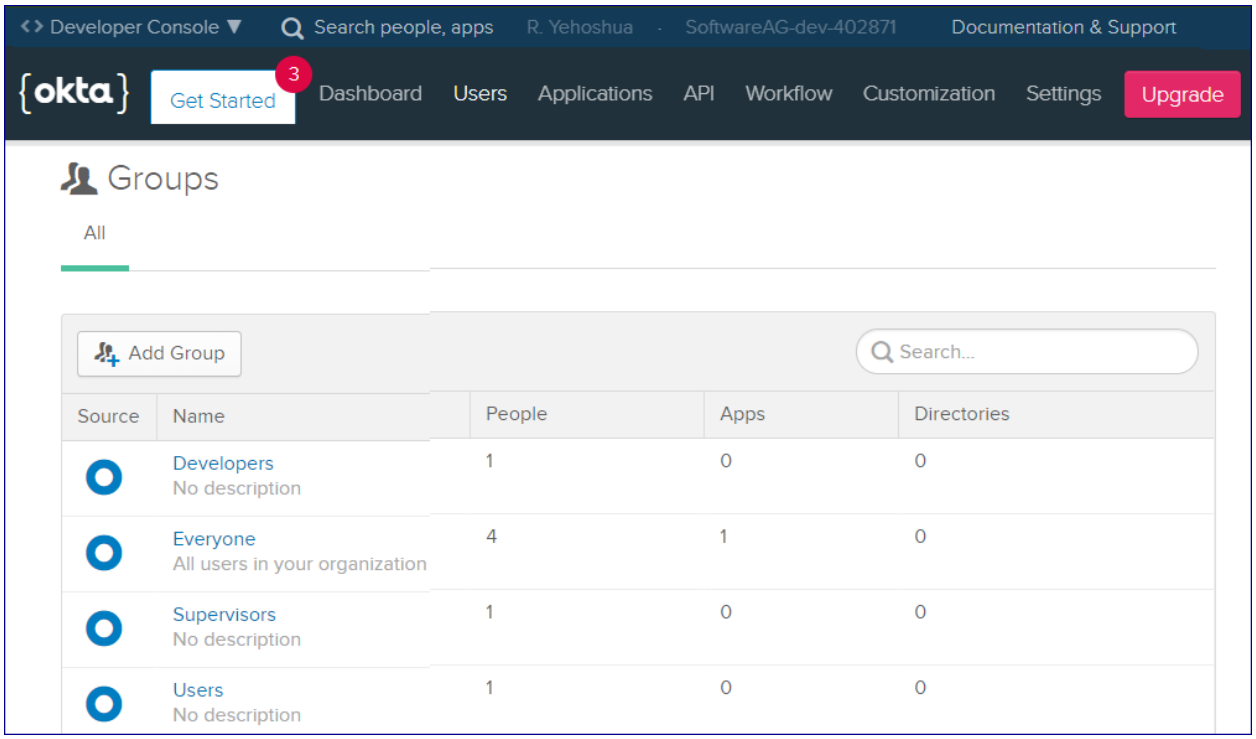
- In the the configuration file *rest-api-config.json* configuration file, set security parameter `auth` to "`oidc`" and provide the OpenID Connect environment variables.

OpenID Connect Environment Variable	Description
`authorization_endpoint`	The client sends the end-user's browser here to request the user's authentication and consent. This endpoint is used in the code and implicit OAuth 2.0 flows that require end-user interaction.
`token_endpoint`	Post an OAuth 2.0 grant (code, refresh token, resource owner password credentials, client credentials) to obtain an ID and/or access token.
`userinfo_endpoint`	Retrieve profile information and other attributes for a logged-in end-user.
`client_secret`	The client secret, which you get from your OpenID Connect provider.
`client_id`	The client ID, which you get from your OpenID Connect provider.
`redirect_uri`	Optional. The `redirect_uri` specified in your ID provider configuration.

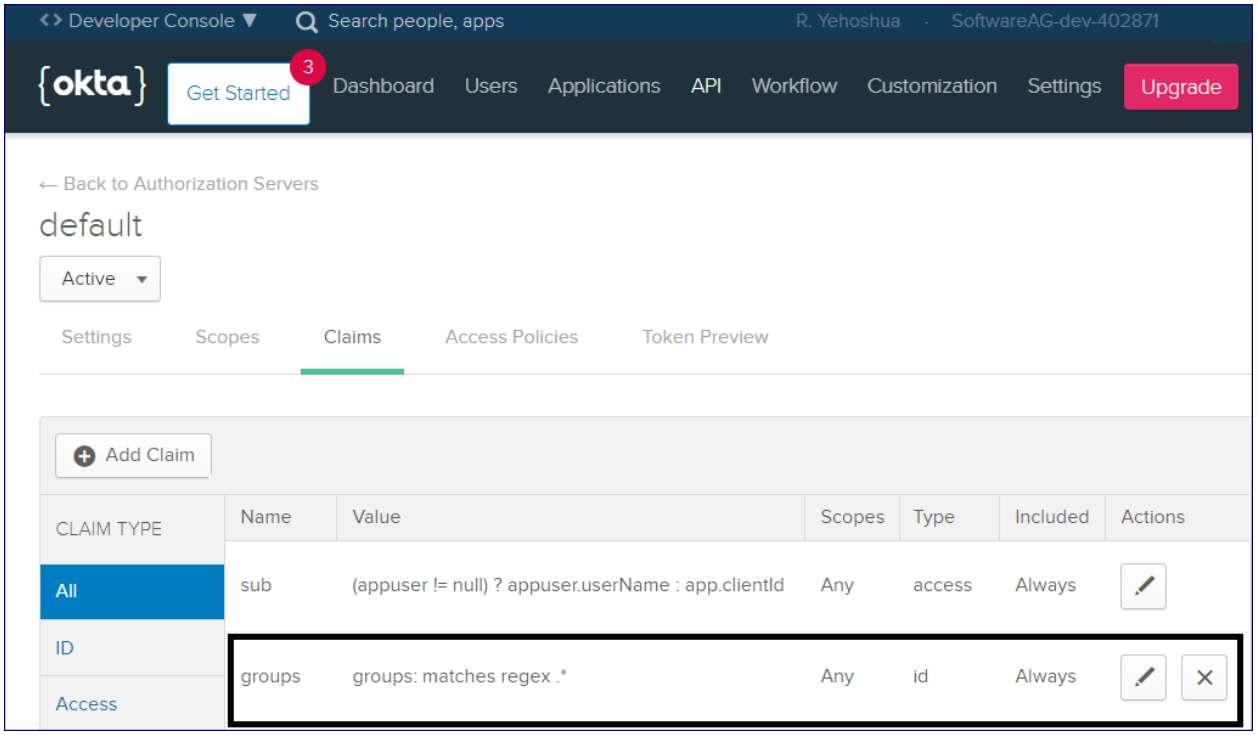
Example using Account Created in Okta

```
{
    "serverURL": "applinx://localhost:2323",
    "applications": ["InstantDemo", "SOADemo", "TestDemo"],
    "security": {
        "auth": "oidc",
        "groups": {
            "admins_group": "Supervisors",
            "developers_group": "Developers",
            "users_group": "Everyone"
        },
        "oidc": {
            "authorization_endpoint":"https://dev-301825.okta.com/oauth2/default/v1/authorize",
            "token_endpoint":"https://dev-301825.okta.com/oauth2/default/v1/token",
            "userinfo_endpoint":"https://dev-301825.okta.com/oauth2/default/v1/userinfo",
            "client_secret": "{YourClientSecret}",
            "client_id": "{YourClientId}"
        }
    }
}
```

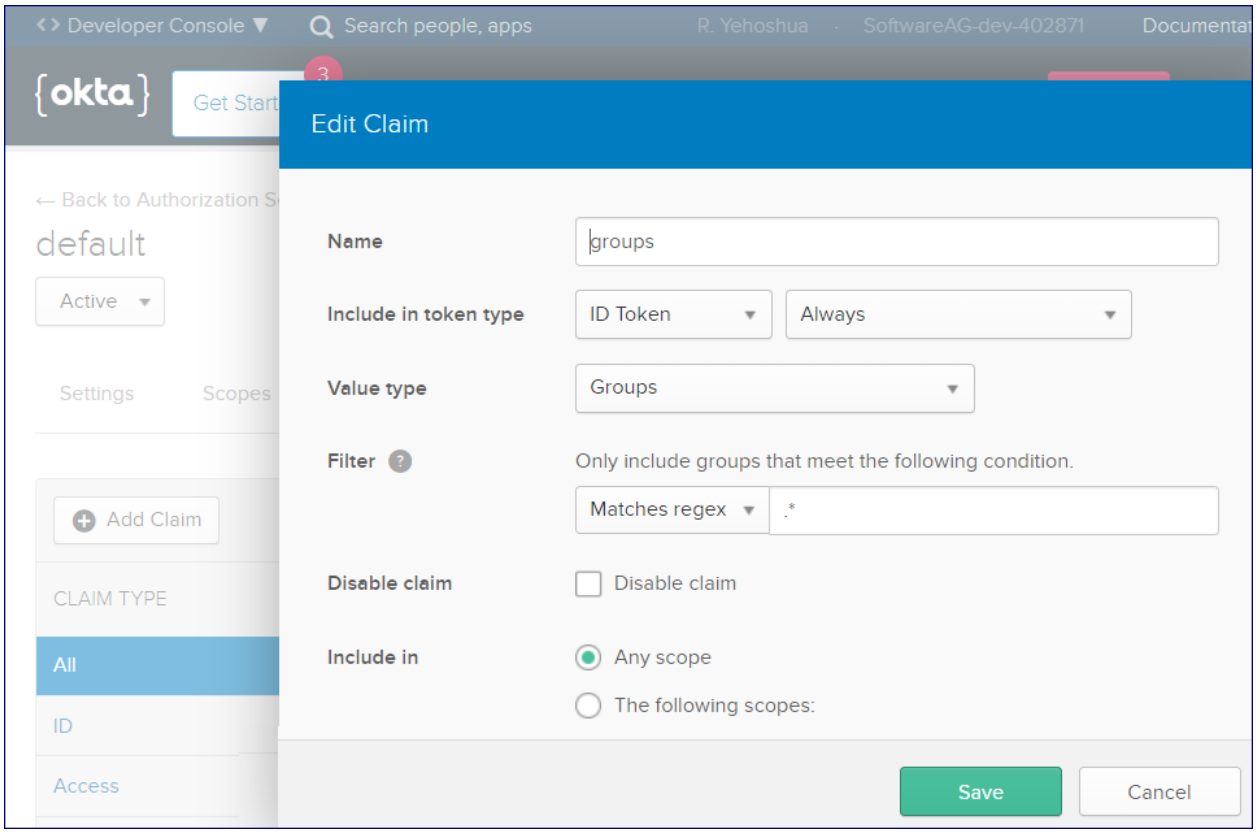
For this example, three groups (Supervisors, Developers, Users) were created in Okta. The Everyone group is available out-of-the-box.



For this example, a custom group `claim` was also created. This will be returned in the ID token. This means that the REST API will know which group the user (who is trying to authenticate) belongs to.



Here is the configuration of the group claim:

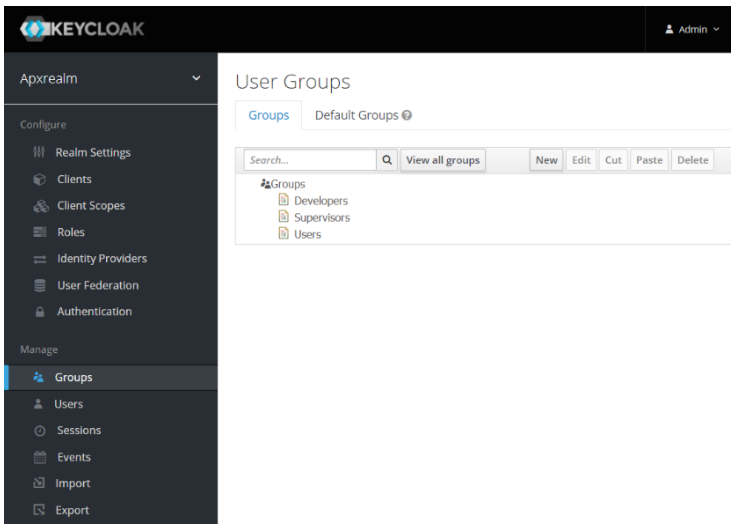


Example using Account on Keycloak

This second example uses an account created on Keycloak. In Keycloak we get the groups out-of-the-box, so there is no need to create a custom groups `claim` we did in the first example. Keycloak configuration:

```
{
    "serverURL": "applinx://localhost:2323",
    "applications": ["InstantDemo", "SOADemo", "TestDemo"],
    "security": {
        "auth": "oidc",
        "groups": {
            "admins_group": "Supervisors",
            "developers_group": "Developers",
            "users_group": "Everyone"
        },
        "oidc": {
            "authorization_endpoint":"http://vmil-apxdkr-ux:3080/auth/realms/apxrealm/protocol/openid-connect/auth",
            "token_endpoint":"http://vmil-apxdkr-ux:3080/auth/realms/apxrealm/protocol/openid-connect/token",
            "userinfo_endpoint":"http://vmil-apxdkr-ux:3080/auth/realms/apxrealm/protocol/openid-connect/userinfo",
            "client_secret": "{YourClientSecret}",
            "client_id": "{YourClientId}",
            "redirect_uri": "http://myAngularApp.com"
        }
    }
}
```

Like the first example, we created three groups here as well:



Connecting an ApplinX Session and Authenticating with OpenID Connect

The flow of a `Create Session Request` when using OpenID Connect authentication is somewhat different.

To create a session

1 Send a regular `POST` request to the session endpoint, for example

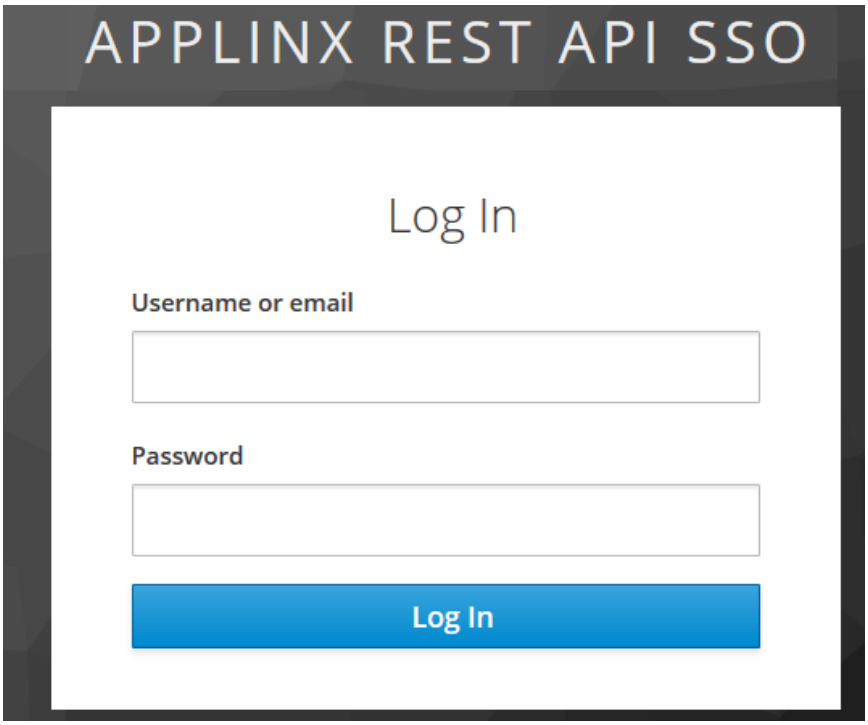
```
http://{host}:{port}/applinx/rest/session
```

2 You will receive a JSON response containing the variable `redirect_uri`. For example:

```
{
    "redirect_uri": "http://vmil-apxdkr-ux:9080/auth/realms/apxrealm/protocol/openid-connect/auth?client_id=r4&
        redirect_uri=http://localhost:8080/api/rest/session&scope=openid&res
}
```

This link will send you to authenticate against your OIDC provider.

3 Enter your credentials and confirm to create a new session, for example:



If your credentials are valid, you will be redirected from the login page to your 'redirect_uri' with a one-time code, which will placed as a query parameter.

4 Send a request to the REST API connect session endpoint with the code as a query parameter and a new session will be connected. You should also send request body that contains the application name to connect to. See *Creating a Session*.

Host Authentication

This authentication mode is used for hosts such as Natural-UNIX that already have an authentication step.

To configure Host Authentication

1 In the the `rest-api-config.json` configuration file, set security parameter `auth` to "`hostAuth`".

2 In the request body, send the host credentials in order to connect a session:

```
"naturalUsername": "string",
"naturalPassword": "string",
```

Disabled

This authentication mode (no authentication) is used for hosts that have a screen containing an authentication step. This mode means you can disable the additional third-party authenticator. This setting is also useful if you do not want authentication before creating a session.

To disable authentication

- In the the `rest-api-config.json` configuration file, set security parameter `auth` to "`disabled`".

Executing a Procedure

For information on procedure entities and procedure groups, see *Procedures*

Introduction

With the REST API you can publish procedures as REST endpoints and then execute the procedures, using an HTTP call with JSON request body containing the Procedure inputs, and a JSON response body representing the Procedure outputs. The procedure I/O outputs are defined in the Designer when you create a procedure.

To publish a procedure as a REST API endpoint

1. Assign the procedure to a Procedure Group. See *Designing and Developing an Application* > *ApplinX Entities* > *Procedure Groups (used as Service Providers)* > *Assigning a Procedure to a Procedure Group*.
2. Configure the application that contains the procedure group in the REST API configuration file (rest-api-config.json) as application in the 'applications' array.
3. Restart the ApplinX server. We recommend you add all your procedures at the same time so you only need to restart to restart your server once.

When a Procedure is published as a REST API endpoint, the OpenAPI documentation of the REST API will contain all the information about the procedure endpoint (Request and response body schema, Authentication schema, HTTP method). See ***Getting Started***

Path Procedure Endpoints

Path Procedures expose two types of endpoints:

- Stateless
- Stateful

A *stateful* endpoint has the suffix '/Stateful'; a *stateless* endpoint does not have a special suffix. Other procedure types (Flow, Web, Program) exposing only one endpoint (stateless). Your Open API documentation will contain documentation about the two endpoints.

Stateless

A stateless procedure execution does not require a session. You can send a procedure call through the REST endpoint, get the output and no session remains open.

To execute a stateless Procedure

- Add your authentication credentials to the request (unless configured with OpenID Connect.

Note: The basic ApplinX and LDAP authentication methods require simple username and password that is sent from the end user to the ApplinX REST API. In OIDC, however, end users do not send their credentials directly to ApplinX. Instead, a third-party ID provider ((Okta, Keycloak, etc...) is used for authentication. A code is sent and then verified in the

ApplinX REST API. If verification is successful, the REST API returns a session token that will be used by the user in all subsequent requests. This token represents a session within the REST API. So when you want to execute a stateless procedure with OpenID Connect you need to connect a session, receive a session token and send the token in the subsequent procedure call request.

Stateful

Stateful procedure execution requires an established session context, the procedure is executed on the session and the session remains connected after the procedure run is finished.

To execute a stateful procedure

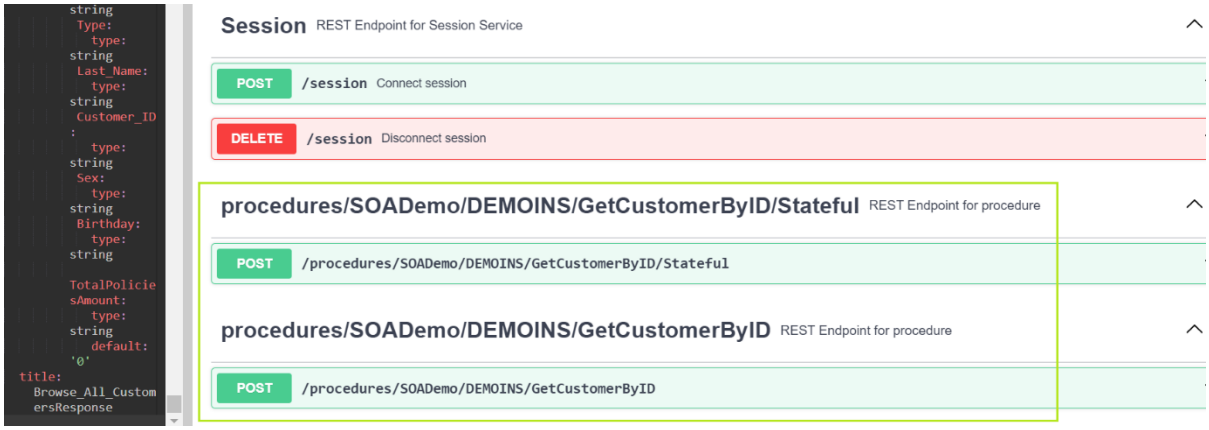
- Add the authorization token you received within the response of the HTTP call to the endpoint of connect session service.

Procedure REST Endpoints Structure

When a Procedure is assigned to a Procedure Group, it is deployed dynamically as a REST service. The endpoint of the REST service is structured according to the name of the application, procedure group and procedure in this exact order. See sample URL:
http://localhost:2380/applinx/rest/procedures/myAppName/myProcedureGroupName/myprocedureName

If the Procedure is of type 'Path Procedure', an additional endpoint is created with suffix '/Stateful'. See sample URL:
http://localhost:2380/applinx/rest/procedures/myAppName/myProcedureGroupName/myprocedureName/Stateful

The following is an example of two published endpoints for Path Procedure in the Swagger Editor:



Notes:

1. This endpoint is created for Procedure '`GetCustomerByID`' within Procedure Group '`DEMOINS`' inside application '`SOADemo`'.
2. Every Procedure endpoint has the prefix '`/procedures`'.

4 Using ApplinX Terminal Emulation Proxy

The ApplinX Terminal Emulation Proxy enables redirecting emulation network traffic to pass through the port configured in this screen (only relevant for AS/400 and Mainframe hosts).

To use the ApplinX Terminal Emulation Proxy

1 In the ApplinX Explorer, right-click on the relevant application and choose **Properties**.

2 Click on the **Host>Terminal Emulation Proxy** node.

3 Check the **Use Terminal Emulation Proxy** check box to enable this option.

4 Enter the port number through which you would like to redirect the emulation network traffic.

5 Click **OK**. ApplinX Server must be restarted for this change to take effect.

Note: When recording using Terminal Emulation Proxy, the "Suppress hidden fields" option is not supported.

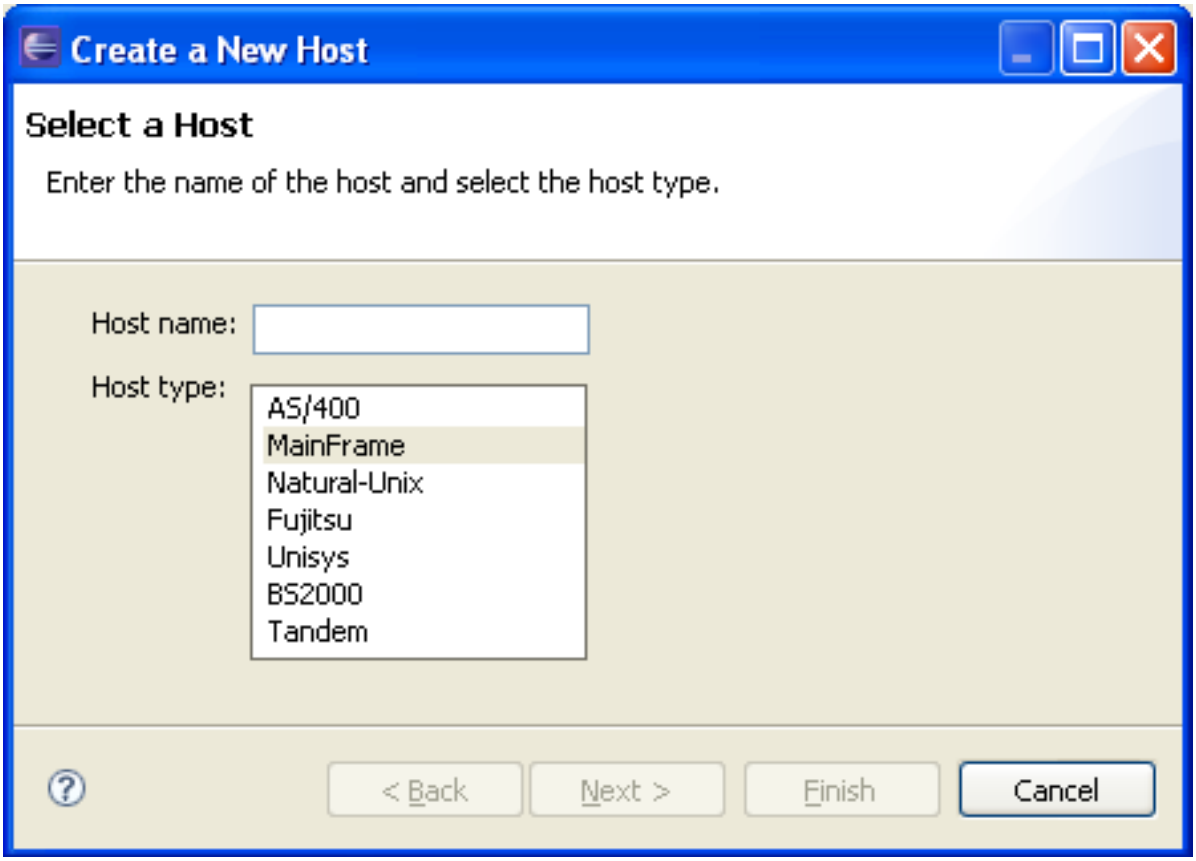
5 Defining Hosts

Defining a New Host

A new host may be defined either in the process of creating a new application (refer to **Creating a New Application**) or simply as an addition host in the host node.

To create a new host

1 In the ApplinX Explorer, right-click on the Host node and choose **New Host**. The New Host wizard is displayed.



2 Enter a name for the host and select a host type. Click **Next**.

3 Set the connection and conversion parameters, which may differ according to the host type. Click **Finish**. Refer to Host Configuration Parameters for further details.

To delete a host, in the ApplinX Explorer, right-click on the relevant host and choose **Delete Host**. Confirm this action.

To edit the current host configuration, in the ApplinX Explorer, right-click on the relevant host and choose **Properties**. The Host Properties' are displayed.

Defining an RPC Host (AS/400 Hosts only)

After defining an AS/400 host, open the host's properties (right-click on the host in the ApplinX Explorer and choose **Properties**), and in the RPC tab configure the parameters.

UNIX Hosts Based on Natural Applications

Standard UNIX protocols lack information required for extensive integration into modern environments. Natural-UNIX, a proprietary protocol of Software AG, provides you with this additional information, and it is therefore recommended to be used with all Natural applications that run on UNIX machines.

Refer to Natural-UNIX Installation for installation details.

To configure a Natural-UNIX host

1 Create and access the Host dialog box as detailed in Adding a Host Definition.

2 Configure the new host: enter a name, IP address (IPv4 and IPv6 address formats are supported), port and service (optional - limited to 24 characters). In the **Device type** field, choose Natural-Unix. Select Natural-APX. Configure the relevant parameters.

There are applications which require entering a user name and password. Click on the **Security** tab. Enter the default **User name** and **Password** (both fields are limited to 24 characters). You can override these settings, either when connecting to a session (in the *Connection Properties* dialog box, **Host user name** and **Host password** fields), or using the Framework/Base Object code or enabling users to dynamically provide a user name and password in the framework login page (refer to the Framework Developer's Guide, General Application Customization, Using a Login Page).

Note: The Application Properties>Windows tab will be disabled, as window's definitions are included in the Natural-UNIX protocol and do not require being defined via ApplinX.

SSH

What is SSH?

Secure Shell (SSH), sometimes known as Secure Socket Shell, is a command interface and protocol for securely getting access to a remote computer. It is widely used by network administrators to control Web and other kinds of servers remotely. SSH commands are encrypted and secure in several ways: both ends of the client/server connection are authenticated using a digital certificate, and passwords are protected by being encrypted. SSH uses RSA public key cryptography for both connection and authentication.

ApplinX supports the SSH Password authentication method. This means the user needs to enter his/her SSH user ID and SSH password in order to establish an SSH connection. This should be done when connecting through the display session configuration, connection pool, Base Object or ApplinX Web Application.

Note: This is applicable only when using a Unisys host.

Configuring an SSH Host

It is possible to configure a host to use SSH connection. In the Host Properties dialog box, Security tab, check the option "Connect using SSH". Change the host port to the host port used for SSH connectivity (the default host port used for SSH connectivity is normally 22).

Uppercase Input

There are fields or screens that their input, whether uppercase or lowercase, is automatically uppercased by the host after receiving it from the user. Therefore, these screens should be configured in ApplinX to have their input fields contents uppercased before being sent to the host, otherwise ApplinX Server will not handle them correctly.

There are three options for doing this:

1. You can add some code for uppercasing the relevant fields before sending them to ApplinX Server.
2. For specific screens, use the following application parameter in the Base Object: GX_APPLICATION_PARM_UPPERCASE_INPUT, "true" (or "false") (For information regarding the usage of application parameters, refer to the Base Object documentation).
3. In the Host Configuration dialog box, check the check box **Uppercase input**. This will convert to uppercase all fields sent to the host. This configuration may be overridden for specific screens using the application parameter described in the previous section.

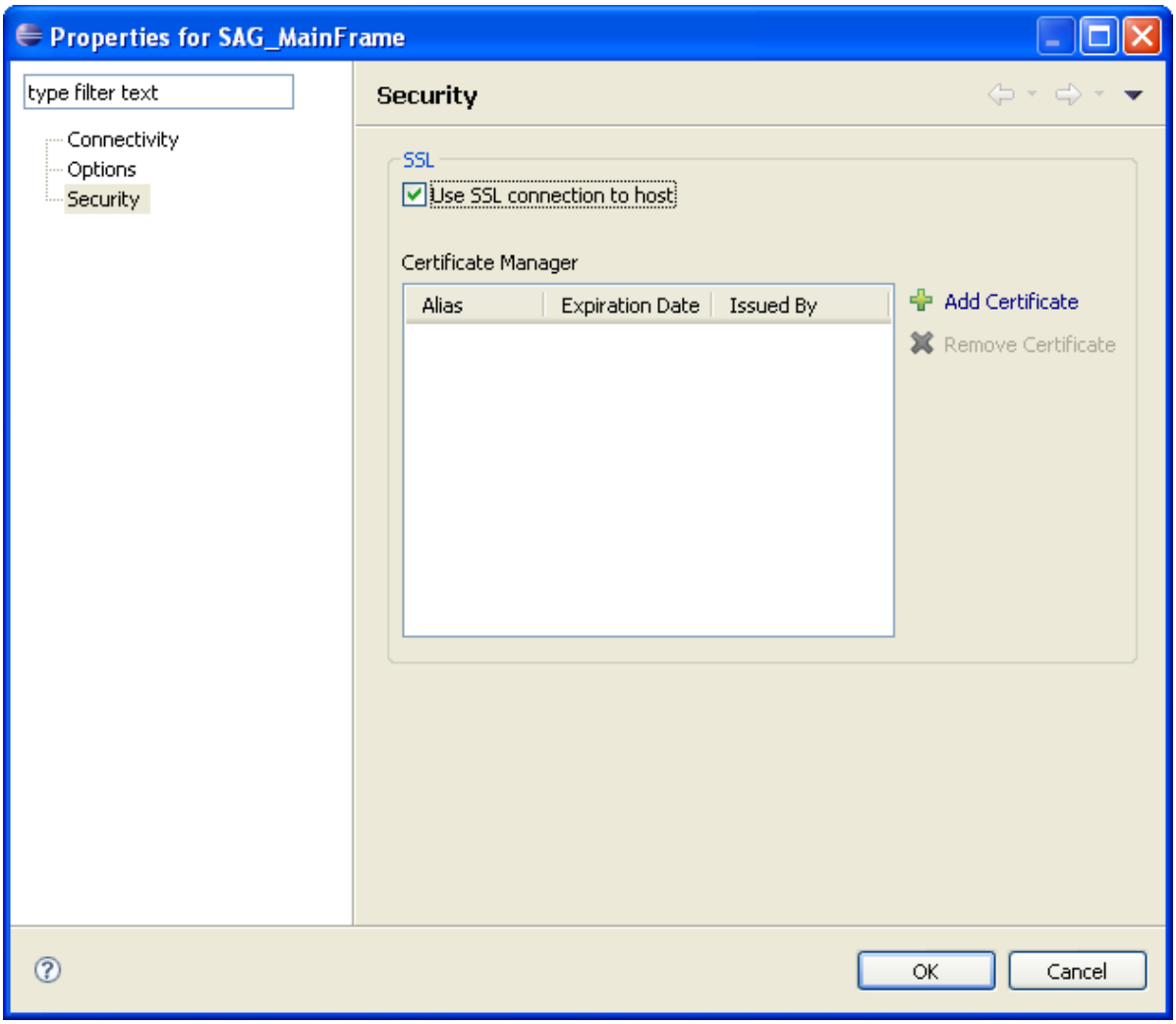
Configuring the SSL Connection

SSL connection is used to ensure a secure connection between the ApplinX server and the remote host. Connecting using SSL V3 enables encrypting all the traffic between the server and the host and requires the server to present proof of identity (server authentication).

Note: This can be used in any block mode host, however this has only been tested on a 3270 Mainframe host.

To define the SSL connection

1 In ApplinX, enable the SSL Connection option for the specific host (access the *Host Properties* dialog box of the relevant host, and in the *Security* tab check the **Use SSL connection to host** check box).



2 Add a valid X509 certificate to connect to the SSL enabled host.

Refer to SSL Cipher Suites Supported by ApplinX.

Defining Host Parameters Required when working with RPC

In the Host Properties dialog box, set the RPC parameters. Refer to RPC Parameters for details.

This feature is available in SOA applications only.

6 Deploying with ApplinX

Deploying an ApplinX Application

When placing the ApplinX application on the target server for the first time, it is necessary to export both the application configuration and the entities. Once the ApplinX application has been created on the target server, it is only necessary to export the entities.

Note: When there are changes in the configuration, you may need to delete the ApplinX application and import it again or manually make the configuration changes on the target server.

To deploy the ApplinX application

1 Export the ApplinX application's configuration and entities (refer to **Exporting an Application Configuration or Entities**). A GXAR file is created in the defined target folder.

2 Import the exported file to the target ApplinX server (refer to **Importing an Application Configuration or Entities**).

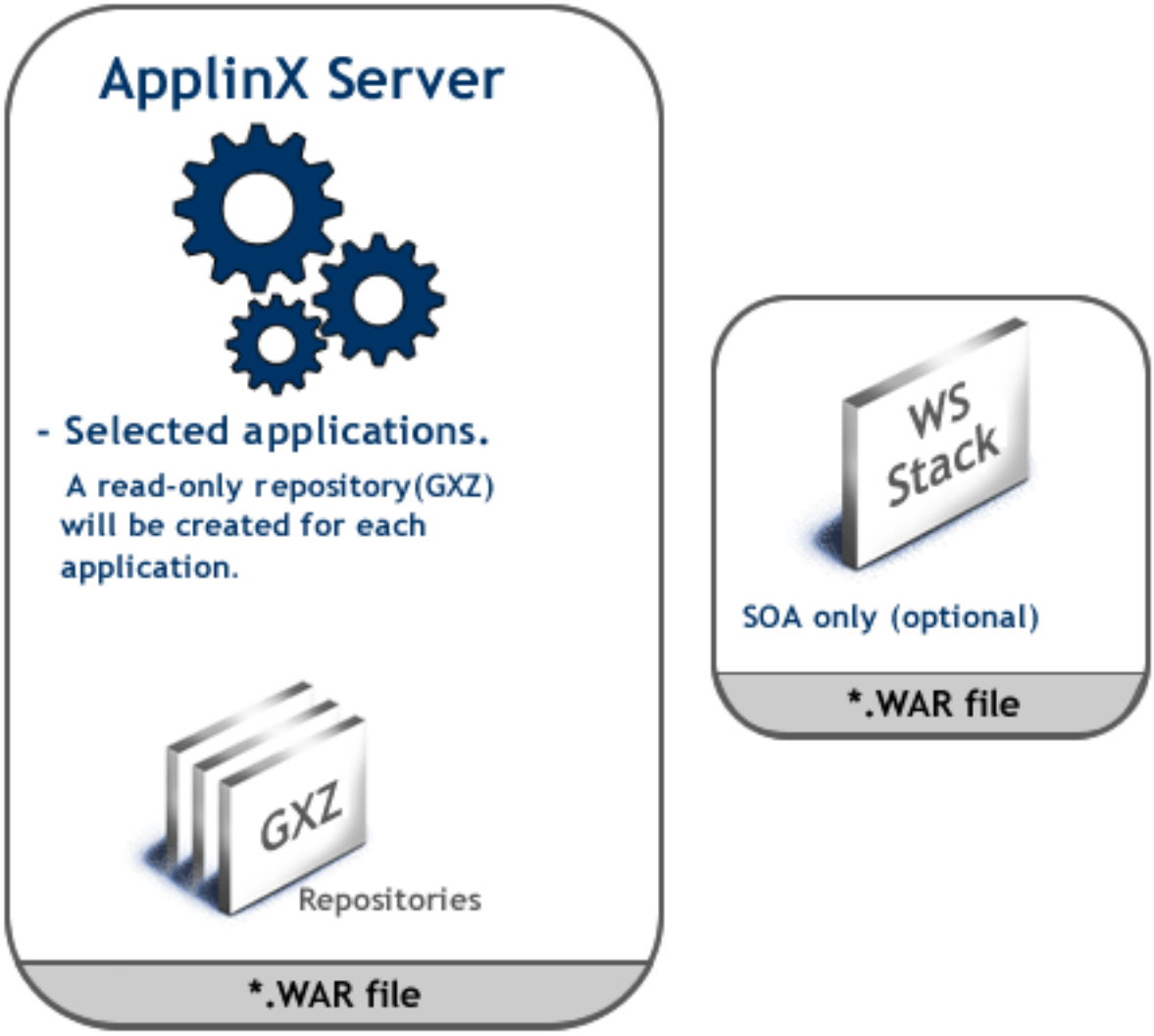
Note: In the production environment it is possible to use the export file as a read only repository.

3 Confirm that the option to trace files is disabled (refer to Application Properties>Host Tab Parameters>Record section). It is recommended not to produce trace files when in production, as this can affect performance.

4 Edit and configure the log settings (refer to Server Properties>Log). It is recommended to set the **Log Level** field to **Warning**.

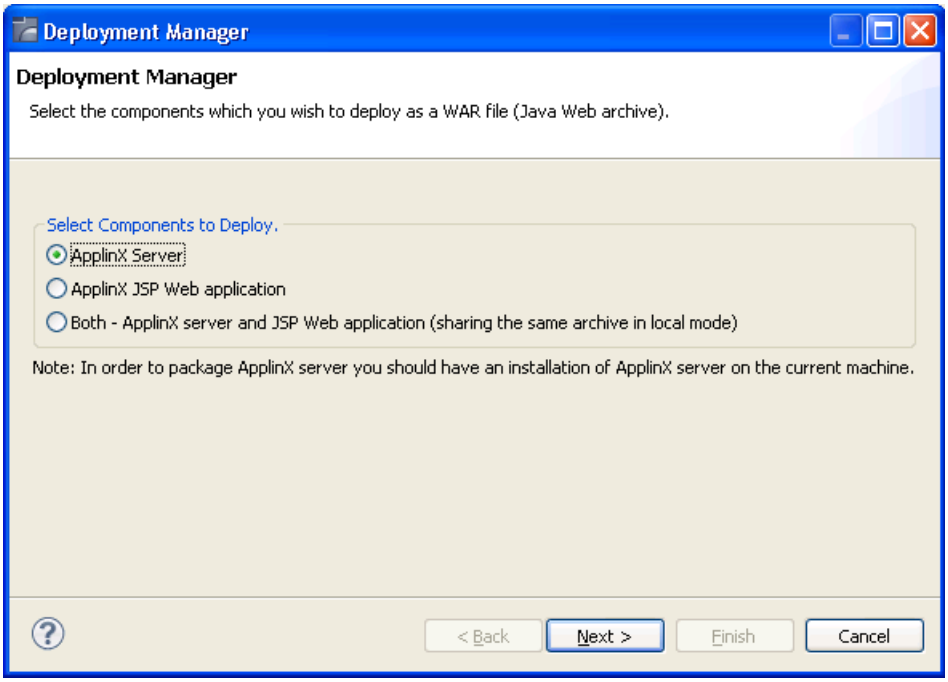
5 When relevant, follow the steps detailed in Deploying an ApplinX Web Application.

Deploying ApplinX Server as a WAR File (Java Web Archive)



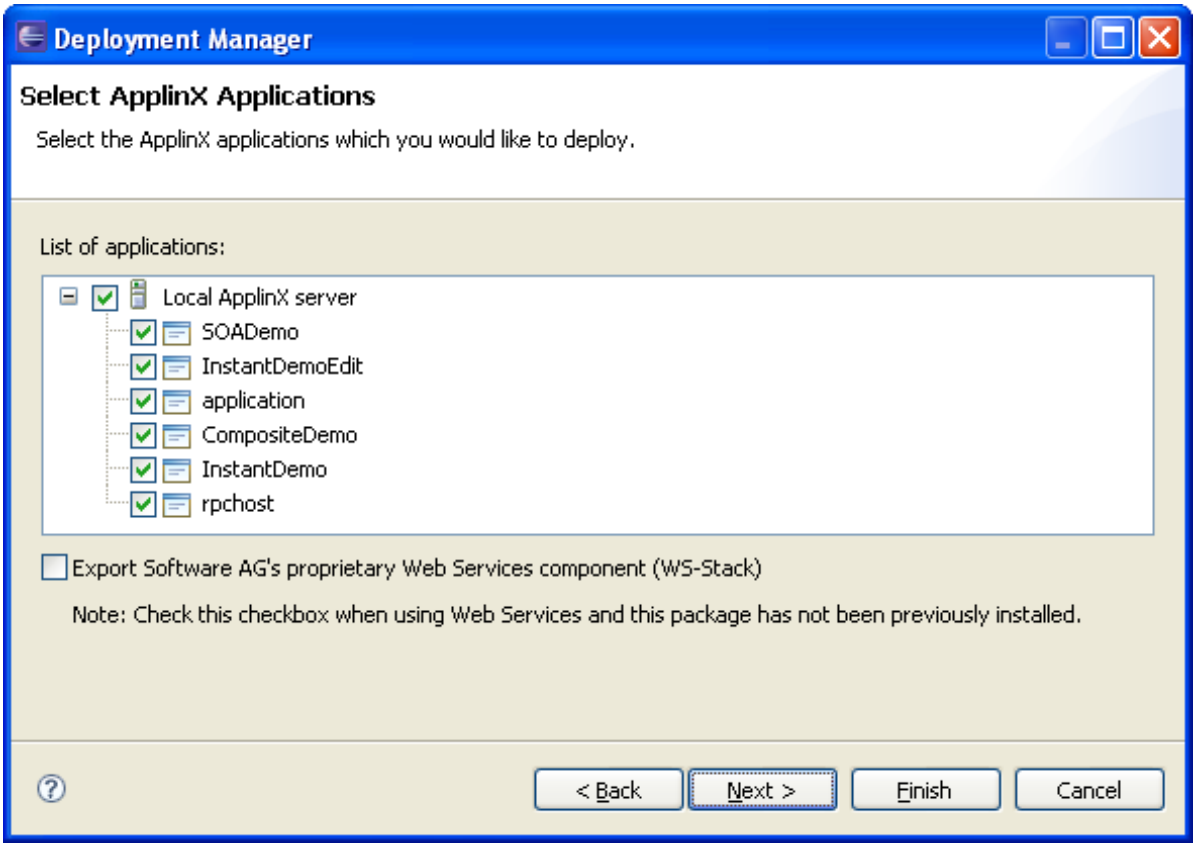
To deploy the ApplinX Server as a WAR File (Java Web Archive)

1 Right-click on the relevant application and choose **Deployment Manager for J2EE...**. The *Deployment Manager wizard* is displayed.



2 Ensure that the **ApplinX server** option is selected.

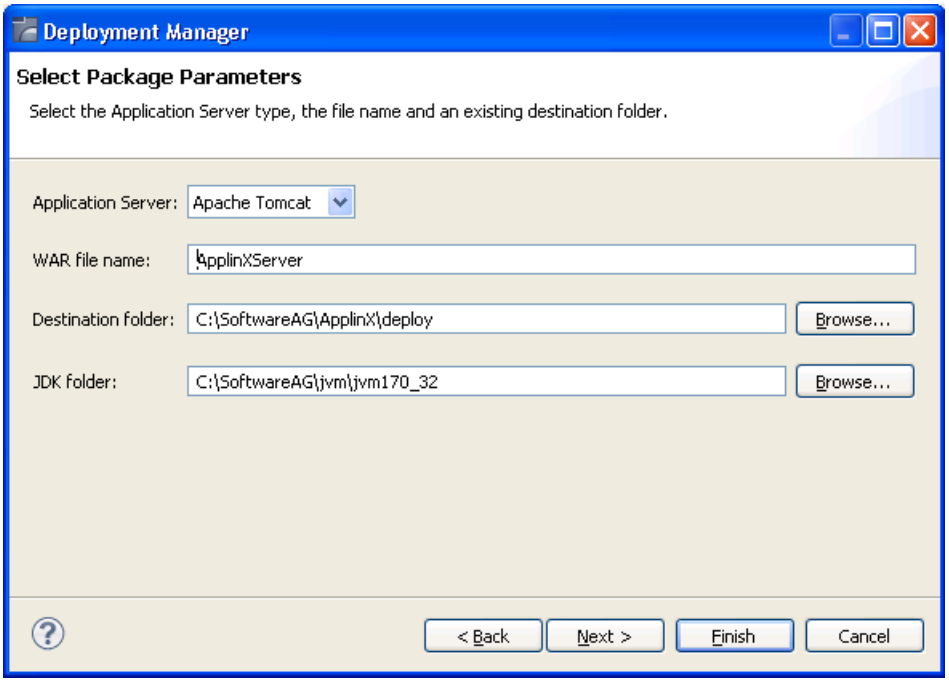
3 Click **Next**. The *Select ApplinX Applications* screen is displayed.



4 Select the Web applications to deploy.

5 Select the Export Software AG's proprietary Web Services component (WS-Stack) when using Web services, and WS-Stack has not already been installed.

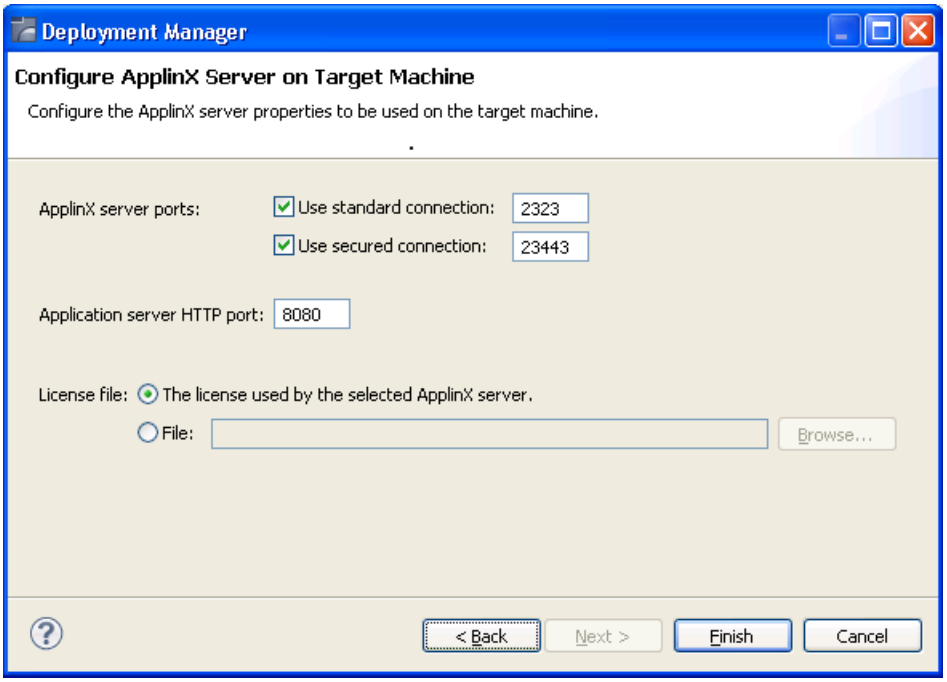
6 Click **Next**. The *Select Package Parameters* screen is displayed.



7 From the list of Application Servers select the server to which you are going to deploy ApplinX server.

8 Enter the WAR file name, the destination folder where the output file will be placed and the JDK folder (ensure that the JAVA_HOME parameter points to the JDK installation folder).

9 Click **Next**. The *Configure ApplinX Server on Target Machine* screen is displayed.

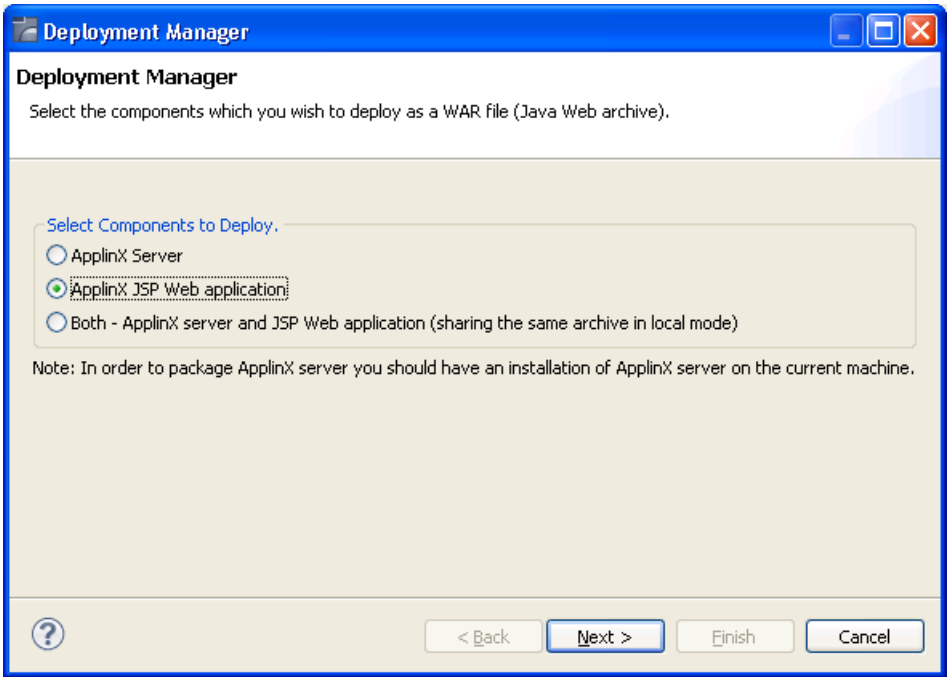


10 In this screen, configure the ApplinX server parameters to be used on the target machine: the ports, and the license file.

Deploying an ApplinX Web Application (JSP) as a WAR File (Java Web Archive)

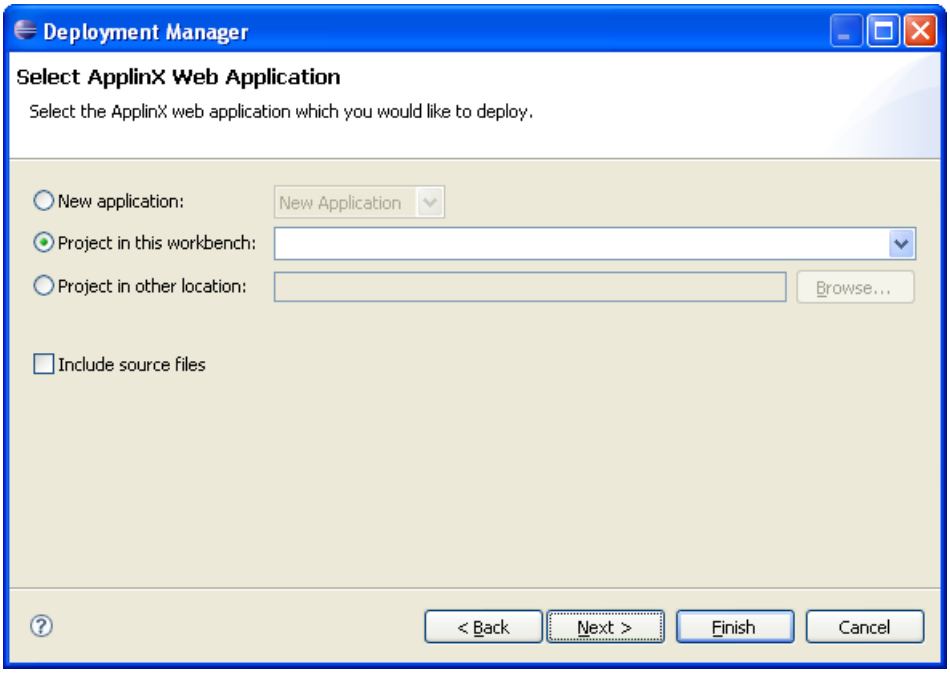
To deploy the Web application:

1 Right-click on the relevant application and choose **Deployment Manager for J2EE...**. The *Deployment Manager wizard* is displayed.



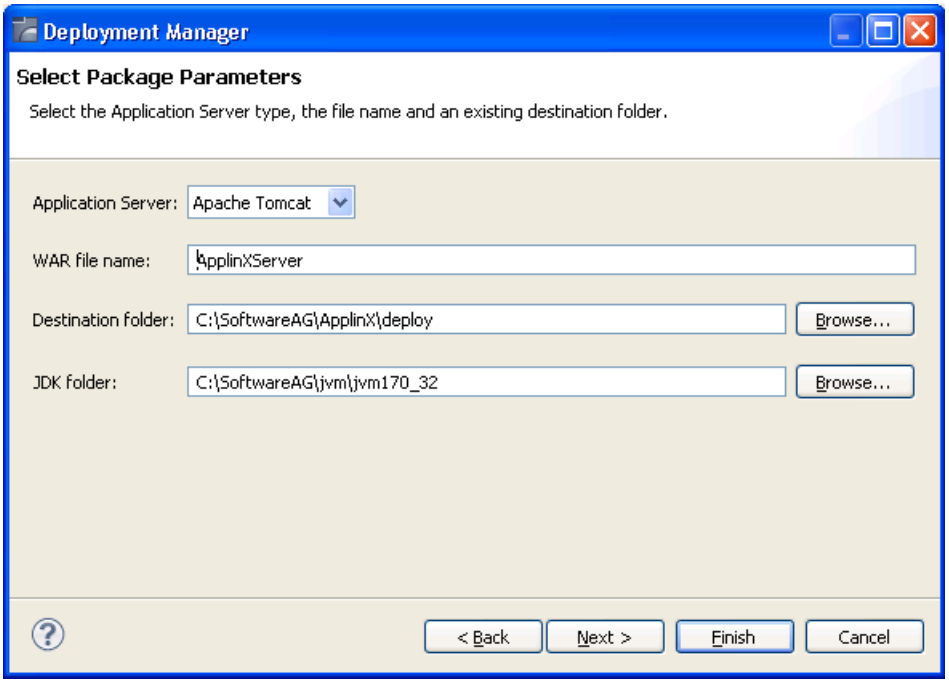
2 Ensure that the **ApplinX for JSP Web application** option is selected.

3 Click Next. The Select ApplinX Application screen is displayed.



4 Select whether to deploy a new application or HTML emulation, or a project from within the workbench, or a project from a different location.

5 Select **Include source files** to include the Java files as well as the compiled classes. Click **Next**. The *Select Package Parameters* screen is displayed.

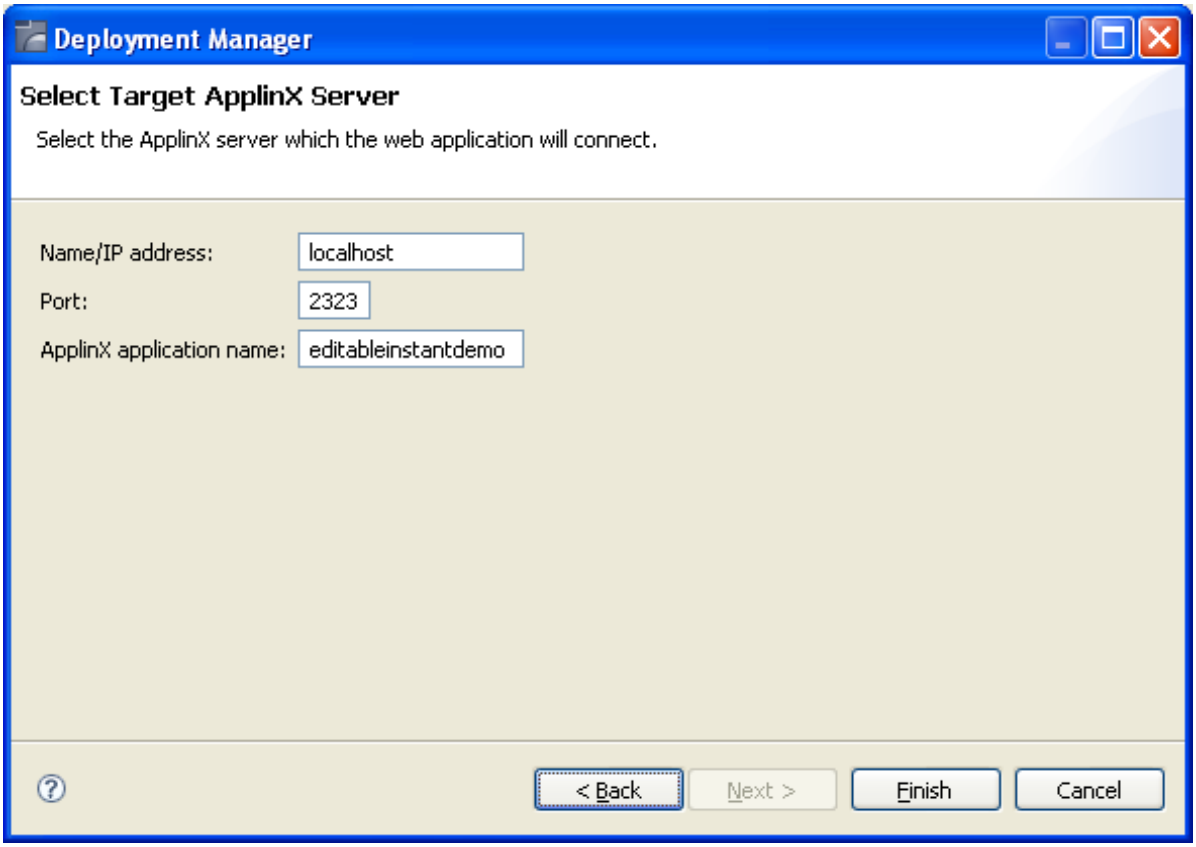


6 From the list of Application Servers, select the server to which you are going to deploy ApplinX server.

Enter the WAR file name, the destination folder where the output file will be placed and the JDK folder (ensure that the JAVA_HOME parameter points to the JDK installation folder).

Note: When selecting WebLogic, you need to manually extract the content of the generated ZIP archive file.

7 Click **Next**. The *Select Target ApplinX Server* screen is displayed.



8 In this screen, enter the ApplinX server and the port with which the Web application will connect. Also enter the ApplinX application name.

9 Click **Finish**.

10 In the console area, it is possible to see whether the process succeeded or failed. The WAR file created should be placed in your Web server directory.

Consider changing the following for production use

1 Change the framework logging according to your needs either by accessing the "Framework Configuration Editor" or in config/gx_logConfig.xml file. For example:

```
<category additivity="false" name="com.sabratec">
          <level value=" ERROR"/> <!-- Sets the log to Errors only -->
           <appender-ref ref="FRAMEWORK_LOG"/>
</category>
```

Possible values are:

- INFO - Normal
- WARN - Warnings

- ERROR - Errors only
- DEBUG - Debug

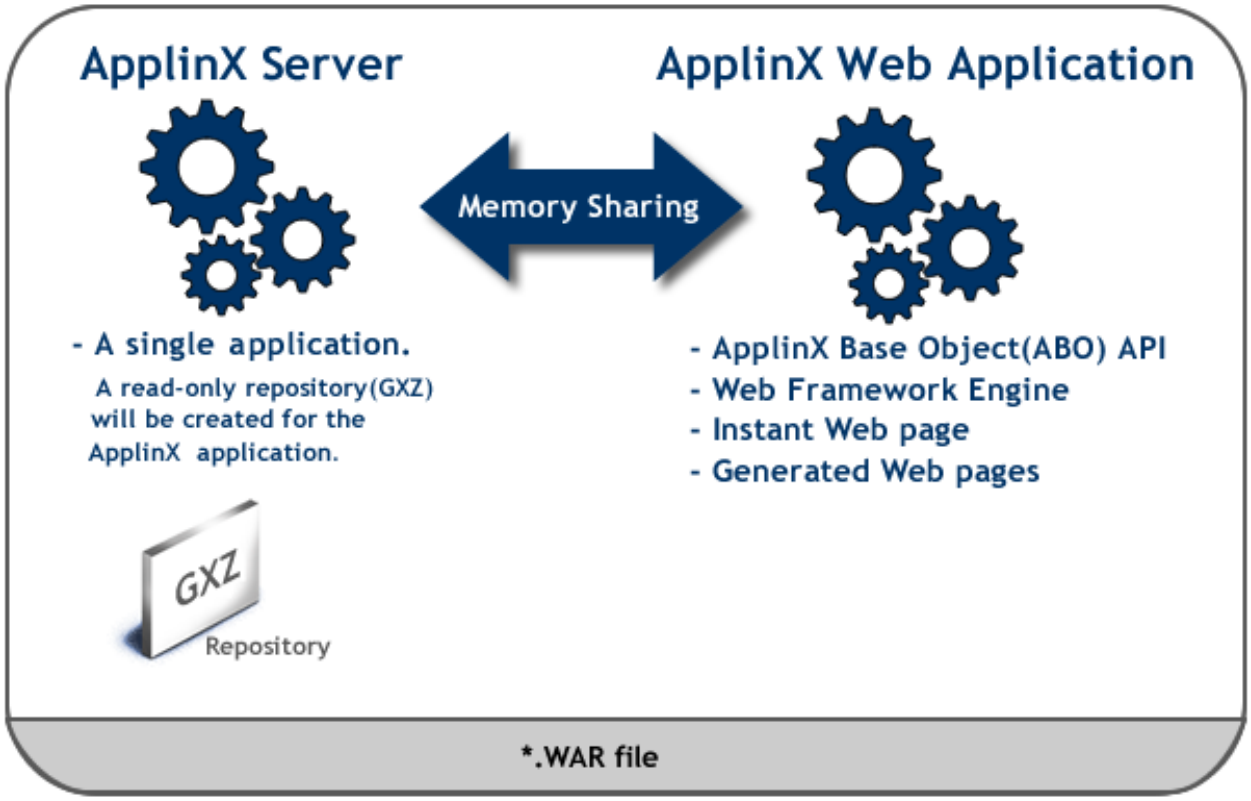
2 Disable the Performance monitor either by accessing the "Framework Configuration Editor" or in config/gx_appConfig.xml file set the WritePerformanceLog to false (this is the default configuration).

3 Disable the Javascript log using the config/gx_clientConfig.xml by setting LogLevel to 0 and ShowLogConsole to false (this is the default configuration).

4 Remove "Framework Configuration Editor" link from index page: delete the folder z_admin and remove the link to "configuration editor" from the index page.

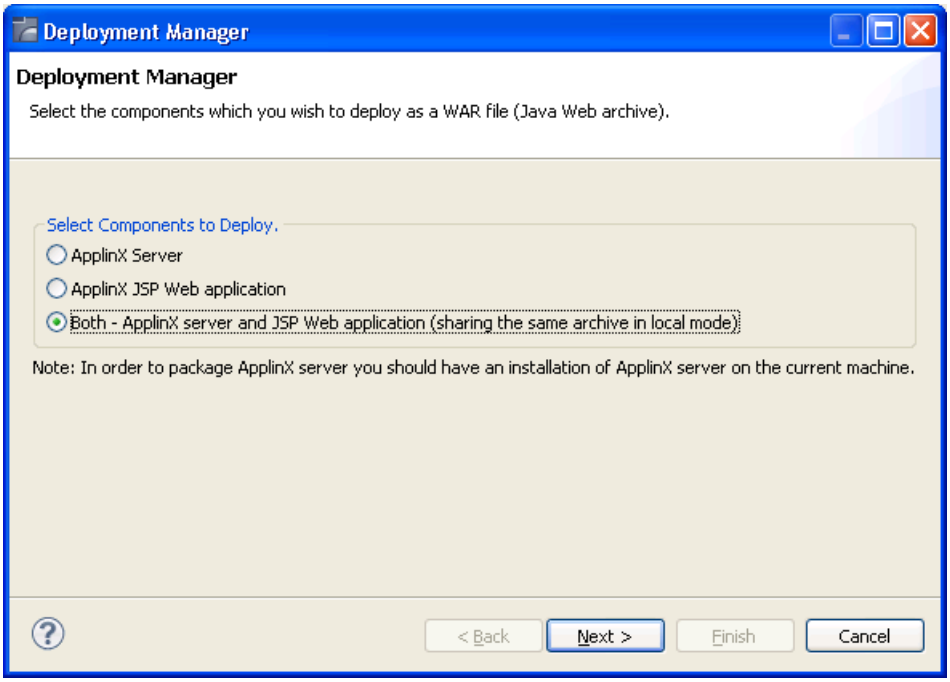
Deploying an ApplinX Web Application (JSP) together with ApplinX Server as a WAR File (Java Web Archive)

This deployment type is typically used when the server and application are to be deployed to the same machine under a Java Application Server. This kind of configuration as the ApplinX server and the Web application share the same memory. Refer to supported application servers within Recommended Software.



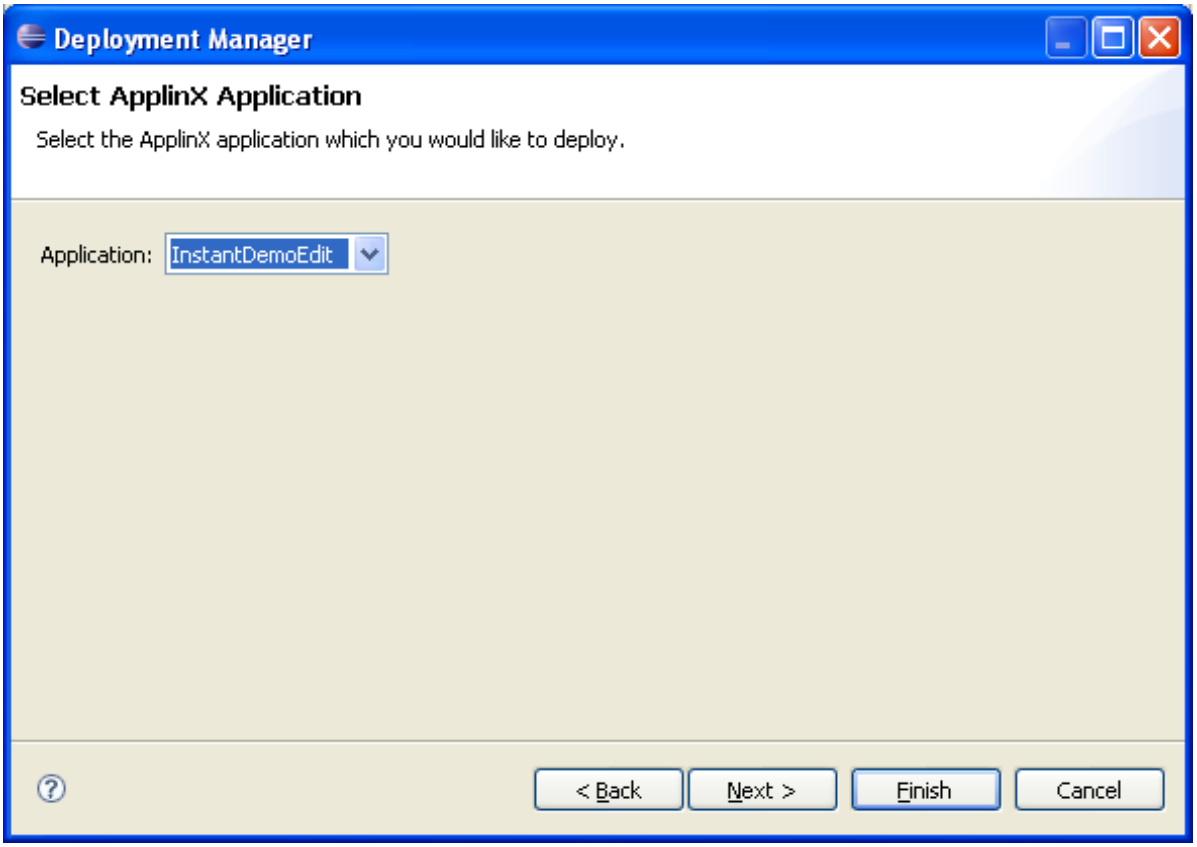
To deploy both the ApplinX server and JSP Web application

1 Right-click on the relevant application and choose **Deployment Manager for J2EE...**. The *Deployment Manager wizard* is displayed.

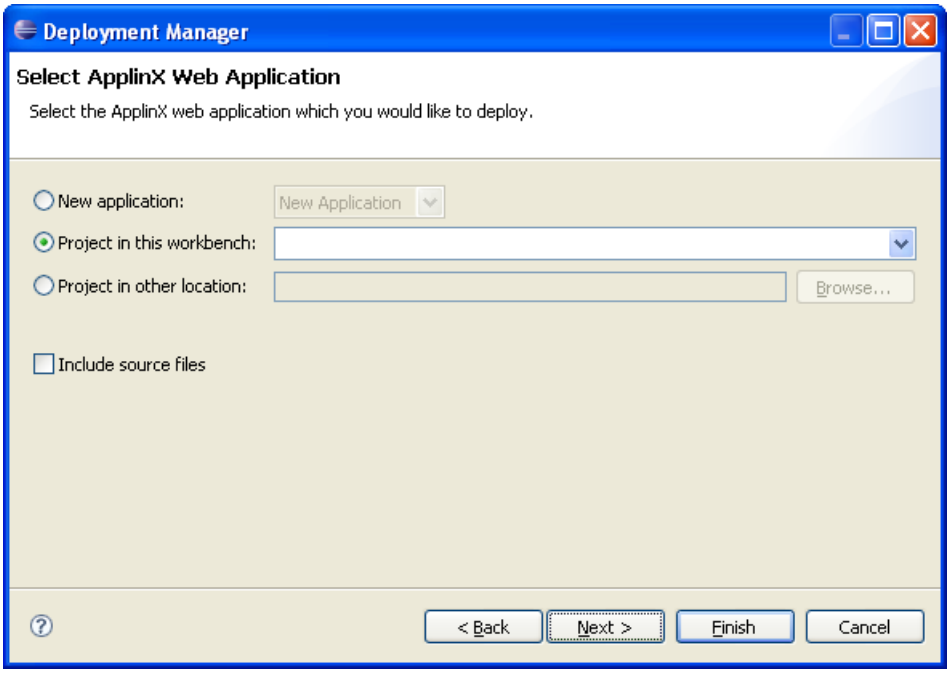


2 Ensure that the **Both - ApplinX server and JSP Web application** option is selected.

3 Click **Next**. The *Select ApplinX Application* screen is displayed.

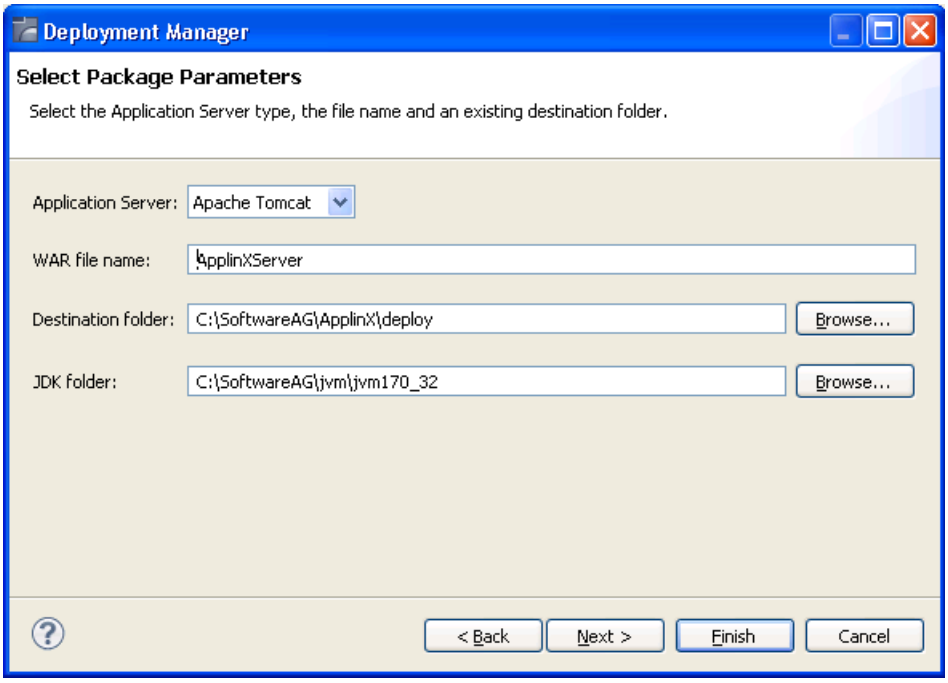


4 Select the ApplinX application to deploy. Click **Next**. The *Select ApplinX Web Application* screen is displayed.



5 Select whether to deploy a new application or HTML emulation, or a project from within the workbench, or a project from a different location.

6 Select **Include source files** to include the Java files as well as the compiled classes. Click **Next**. The *Select Package Parameters* screen is displayed.

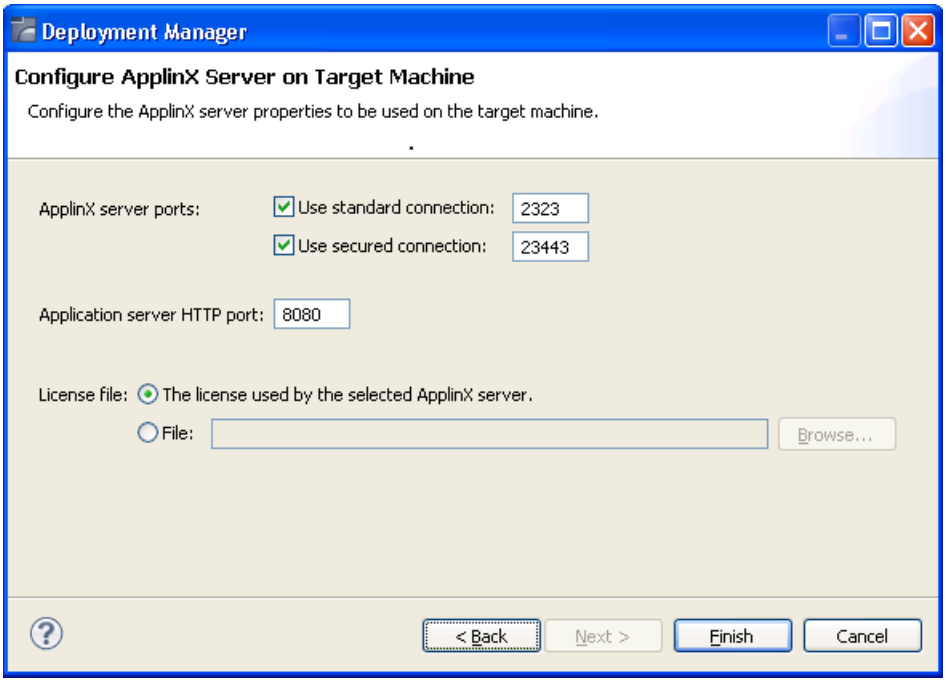


7 From the list of Application Servers select the server to which you are going to deploy ApplinX server.

Enter the WAR file name, the destination folder where the output file will be placed and the JDK folder (ensure that the JAVA_HOME parameter points to the JDK installation folder).

Note: When selecting WebLogic, you need to manually extract the content of the generated ZIP archive file.

8 Click **Next**. The *Configure ApplinX Server on Target Machine* screen is displayed.



9 In this screen, configure the ApplinX server parameters to be used on the target machine: the ports, and the license file.

10 Click **Finish**.

11 In the console area, it is possible to see whether the process succeeded or failed. The WAR file created should be deployed to your Java Application Server.

Consider changing the following for production use

1 Change the framework logging according to your needs either by accessing the "Framework Configuration Editor" or in config/gx_logConfig.xml file. For example:

```
<category additivity="false" name="com.sabratec">
          <level value=" ERROR"/> <!-- Sets the log to Errors only -->
           <appender-ref ref="FRAMEWORK_LOG"/>
</category>
```

Possible values are:

- INFO - Normal
- WARN - Warnings
- ERROR - Errors only
- DEBUG - Debug

2 Disable the Performance monitor either by accessing the "Framework Configuration Editor" or in config/gx_appConfig.xml file set the WritePerformanceLog to false (this is the default configuration).

3 Disable the Javascript log using the config/gx_clientConfig.xml by setting LogLevel to 0 and ShowLogConsole to false (this is the default configuration).

4 Remove "Framework Configuration Editor" link from index page: delete the folder z_admin and remove the link to "configuration editor" from the index page.

Note: When connecting to an external application server (any server that is not Tomcat), the changes performed in the Designer will not be saved.

Deploying an ApplinX Web Application (.NET)

Note: ApplinX framework supports .NET clustering in a Web farm environment.

To deploy the Web application

1 Copy the entire Web application into a temporary working folder.

2 In the *config/gx_logConfig.xml* file edit the Logger settings according to your needs (enable/disable the log, target path and log level). For example:

```
<logger name="com.sabratec">
          <level value="INFO"/> <!"sets the log to normal mode
          <appender-ref ref="FRAMEWORK_LOG"/>
</logger>
```

When you do not require the performance log, ensure that the last two category nodes in the XML file are commented.

3 Change the definition of `designMode` in *config/gx_appConfig.xml* to "false".

4 Change the definition of the serverURL and applicationName in *config/gx_appConfig.xml* to the target server.

5 Copy the updated folder to the relevant Web server.

6 Map this folder as a virtual directory under the server.

Deploying and Running an ApplinX Application on IIS

Note: See *Software Requirements* for supported versions.

The following pre-installation steps are only required the first time.

1. Download and install the Microsoft Visual J# 2.0 Redistributable Package, Second Edition (X64). You can find it under *http://www.microsoft.com/en-us/download/confirmation.aspx?id=15468*.
2. Open the Windows **Add/Remove Programs** on Windows 7 or the Server roles on the Windows Server and select the web server (IIS) component and add to the server the following components:
   a. Security:

      Basic Authentication
      Client Certificate Mapping Authentication
      Digest Authentication
      IP security
      Request Filtering
      IIS Client Certificate Mapping Authentication
      URL Authorization
      Windows Authentication

   b. Application Development Features (Web-App-Development Component)

      .NET Extensibility
      ASP
      ASP.NET
      CGI
      ISAPI Extensions
      ISAPI Filters
      Server Side Includes

   c. Common HTTP Features (Web-Common-Http Component)

      Static Content
      Default Document
      Directory Browsing
      HTTP Errors
3. From Windows **Add/Remove Features**, also add the Microsoft .NET Framework. See *Software Requirements* for supported versions.

After deploying the ApplinX application on the IIS server using the ApplinX Web manager, perform the following steps:

1. Open the IIS Manager

2. Under **Site > Default Web Site**, convert the virtual directory created by the ApplinX Web Manager to Application.
3. Change the Application pool to Classic .NET AppPool.
4. Add to the default documents section the index.aspx page.
5. On the security page, add write-permission to IIS users.

7 Publishing an Application to CentraSite

ApplinX enables publishing an ApplinX application to CentraSite ActiveSOA. The published application will include the ApplinX assets: the Server, application, host, screens, screen groups, screen images, path and flow procedures, procedure groups, external web services and program procedures and the connections ("Associations") between them. The ApplinX assets will then be available to other CentraSite users for impact analysis.

In order to use this feature, ensure that you have a CentraSite license and ensure that the ApplinX and CentraSite are of the same webMethods suite version. CentraSite must be installed on the same machine as the ApplinX server.

Configure CentraSite in ApplinX

To register an application to CentraSite you need to configure the CentraSite connection and enable connectivity to CentraSite.

To enable connectivity to CentraSite

1 In ApplinX Designer, either right-click on the server (in the ApplinX Explorer) and choose **Properties**, or choose **Properties** from the **Server** menu.

The *Server Properties* dialog box is displayed.

2 In the CentraSite tab, choose **Enable CentraSite**. This is available only when you have the required Software AG common files. If the common files are not installed, run the Software AG Installer and within "Infrastructure>Libraries" select to install "Shared Libraries" and "CentraSite Libraries".

3 Enter the name of the host where CentraSite is installed, the port number used to connect to CentraSite, the current user's name and password.

4 Click on the **Test Connectivity** button to test that ApplinX was able to connect to CentraSite.

Register the Application to CentraSite

To register the application

1 In ApplinX Designer, either right-click on the application (in the ApplinX Explorer) and choose **Register Application to CentraSite**, or choose **Register Application to CentraSite** from the **Application** menu.

2 A progress bar is displayed indicating the progress of the process. Once the process is completed, the details are written in the Console area in the Designer.

Update the Application to CentraSite

To update the application

1 In ApplinX Designer, either right-click on the application (in the ApplinX Explorer) and choose **Update Application to CentraSite**, or choose **Update Application to CentraSite** from the **Application** menu.

2 A progress bar is displayed indicating the progress of the process. Once the process is completed, the details are written in the Console area in the Designer. The update operation updates the changes made to the ApplinX assets and associations.

Unregister the Application from CentraSite

To unregister the application

1 In ApplinX Designer, either right-click on the application (in the ApplinX Explorer) and choose **Unregister Application from CentraSite**, or choose **Unregister Application from CentraSite** from the **Application** menu.

2 A progress bar is displayed indicating the progress of the process. Once the process is completed, the details are written in the Console area in the Designer.

8 Log Files in ApplinX

ApplinX provides a total of seven log files, four on the server side and three on the web application side. These logs are based on Log4j and as such are highly customizable. ApplinX log files have multiples uses:

- For an application in production, log files are an important instrument for administrators.
- Log files are also important for problem analysis, debugging, and performance analysis.

Log File Format

Usually, there are two servers where a log is produced:

- the ApplinX server
- the application server hosting the ApplinX web application (e.g. Tomcat)

Log files are plain ANSI text files. Most of the log files in ApplinX are created with Log4J, which means you can customize logging in ApplinX with standard Log4J configuration files. These are the two most important:

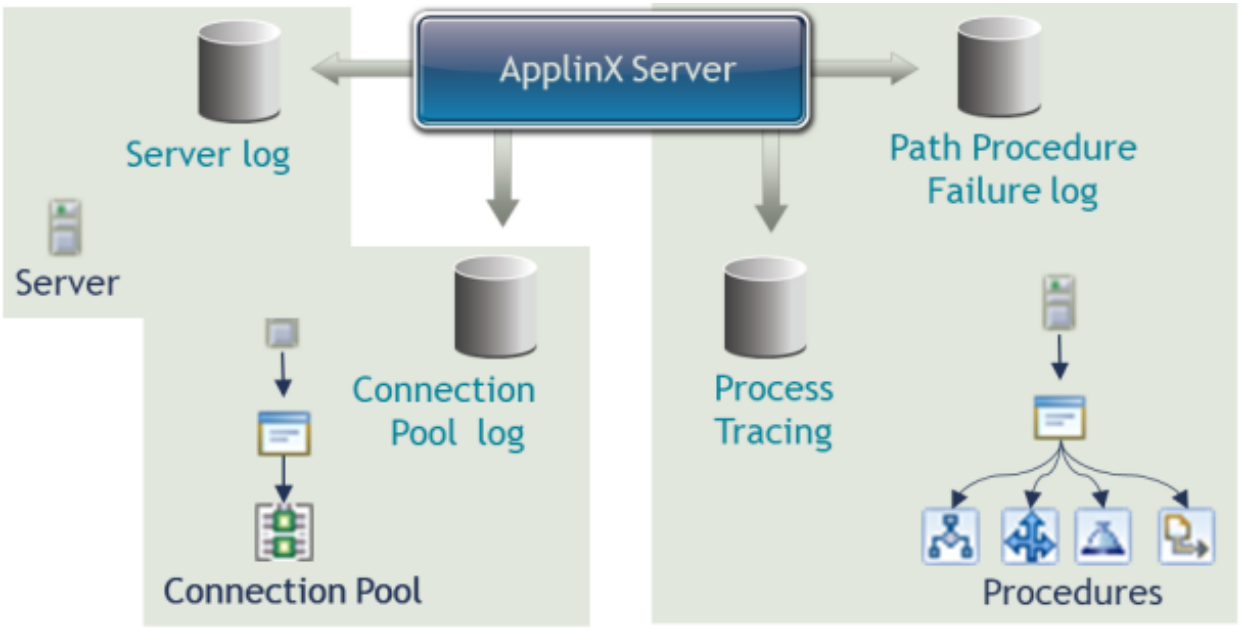
- **gxlog_config.xml**
  for the ApplinX Server side, see ***Log Files on the ApplinX Server Side***
- **gx_logConfig.xml**
  for the Web application side, see ***Log Files on the Web Application Side***

Log Files on the ApplinX Server Side

This section covers the following topics:

See also *Viewing Server Logs* in the Administration documentation.

Scope

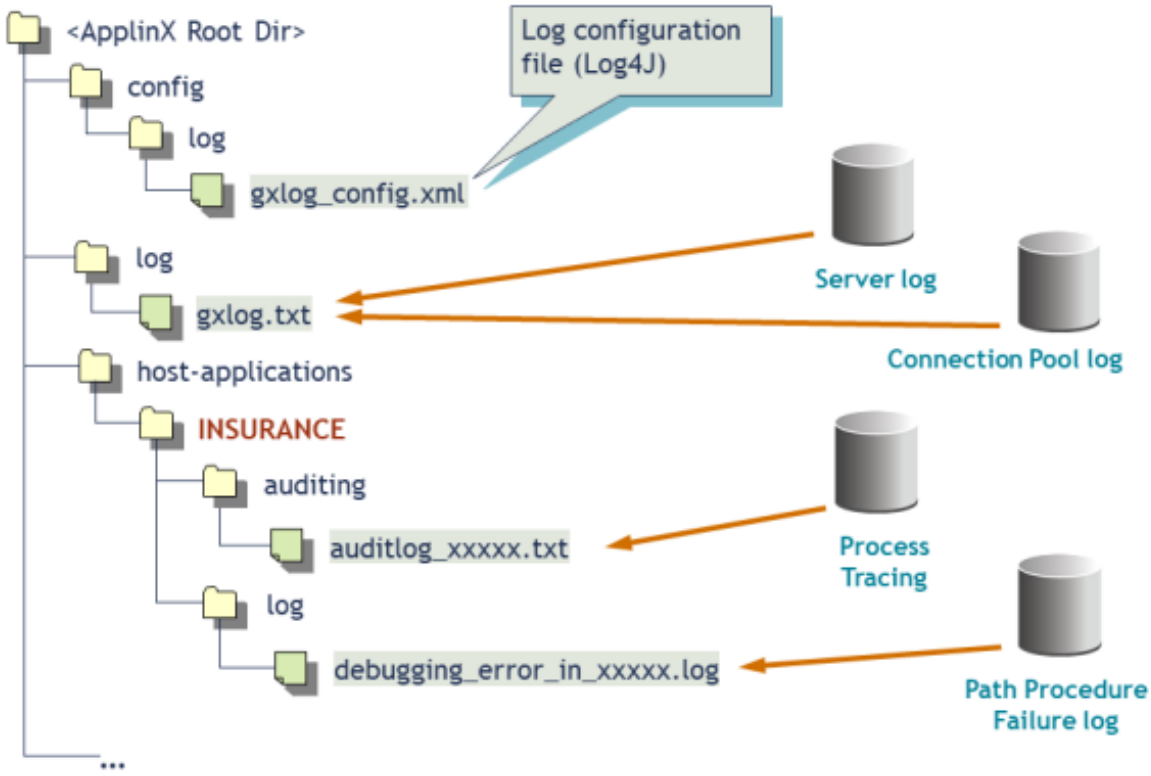


There are four log files on the server side:

- **Server Log**
  This is the main log file of the ApplinX Server. It includes general log data for all applications available on the server (e.g. server startup and shutdown, application loading, session start and end). Its log level can be configured on the level of the ApplinX Server.
- **Connection Pool Log**
  This adds log data specific to Connection Pools to the same file as the Server log. It can be enabled or disabled on the level of a single Connection Pool of an application.
- **Process Tracing**
  This is about performance (it shows execution times of procedures), and is enabled or disabled on the level of a single application. It adds log data specific to procedures (path, flow, etc.).
- **Path Procedure Failure log**
  This can be enabled on the level of the ApplinX Server to obtain a dump of a failing procedure.

Log File Locations

The following graphic shows the location of the log configuration file and the individual log files.



- **Log Configuration File**
  File *gxlog_config.xml* is a standard Log4J configuration file located in the *config/log* folder inside the ApplinX root installation directory.
- **Server log File**
  By default this file is called *gxlog.txt* and is located in the *log* subfolder of the ApplinX root directory.
- **Connection Pool Log**
  This file adds entries specific to Connection Pools to the same file, that is, *gxlog.txt*.

The other two log files are created per application:

- **Process Tracing File**
  You can specify the name of this file when you enable process tracing. Its format format can be .txt or .csv. It's name may include time (%t) or the process ID (%p). By default, it is created in the *auditing* subfolder of the application's directory. In our example, in the *auditing* subfolder of the *host-applications/INSURANCE* folder.

- **Path Procedure Failure Log**
  When enabled, this file is named *debugging_error_in_xxxxx.log* by default and includes the name of the failing procedure, the session ID and the time. By default, it is located in the "log" subfolder of the application's directory.

Names and locations of all log files can be customized in the log configuration file *gxlog_config.xml*.

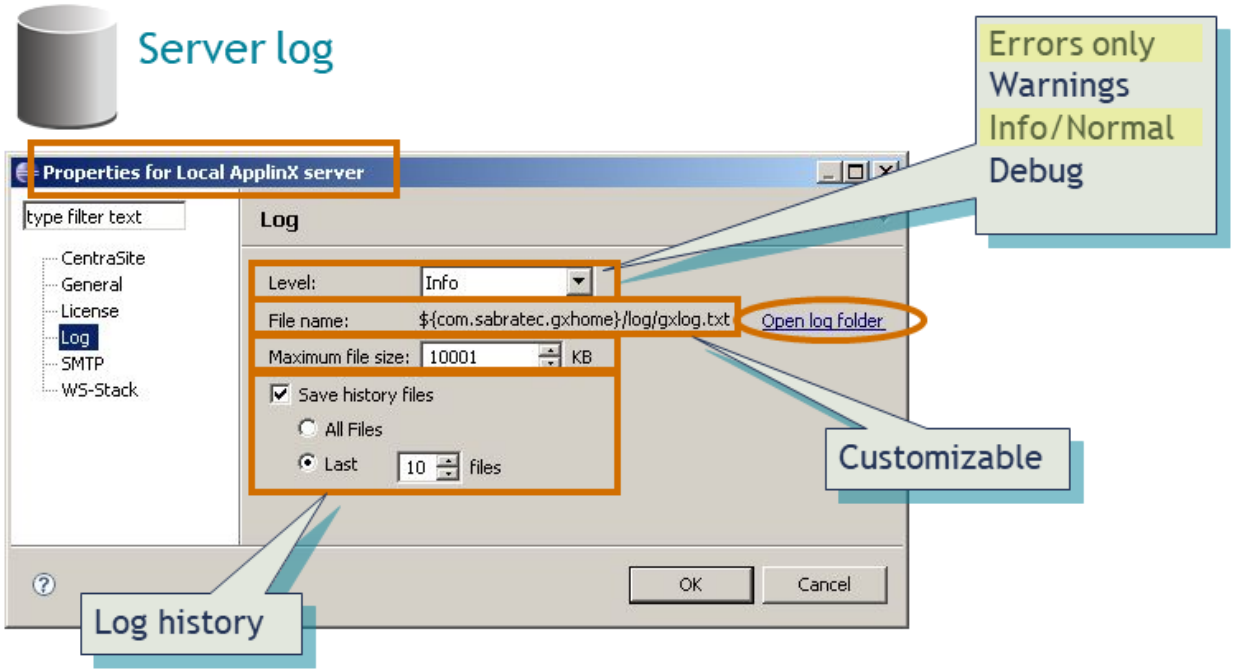
Server Log

The Server log is the main log file of the ApplinX Server. It supports multiple log levels and log history.

- Configuring the Server Log

- Server Log and Connection Pool Log Example

Configuring the Server Log



To configure the Server log file

1 In ApplinX Designer, go to the properties of the server (for example "Local ApplinX server"), and select the **Log** node.

2 You can change the following settings:

Setting	Description
Level	The contents of the log file are as detailed as this property defines, where every level includes the levels above it. For example: Debug level also includes Normal, Warnings and Errors Only levels. By default, the level is "Info/Normal". For an application in production, log level "Errors only" is recommended.
File name	The log is written to the file displayed here. File name and location can be customized in the Log configuration file (gxlog_config.xml).
Open log folder	Enables you to see the list of existing log files.
Maximum file size	Starts a new log file after the current file has been filled to the maximum file size.
Save history files	■ Determines whether backups of old log files will be saved, after restarting the server.

Setting	Description
	■ Determines the number of backups saved before overwriting the old log files. For example: 10 means "save the last 10 log files, in addition to the current one, then start to overwrite". If you select **All Files**, old log files are never deleted. Default value is "10".

3 Restart the ApplinX server for your changes to take effect.

Server Log and Connection Pool Log Example

This example shows a Server log with log level "Info". It shows Server startup logging at the beginning. It also shows a Warning about a wait condition that timed out (e.g. a path waits for a specific screen ID), and an error of a failing procedure.

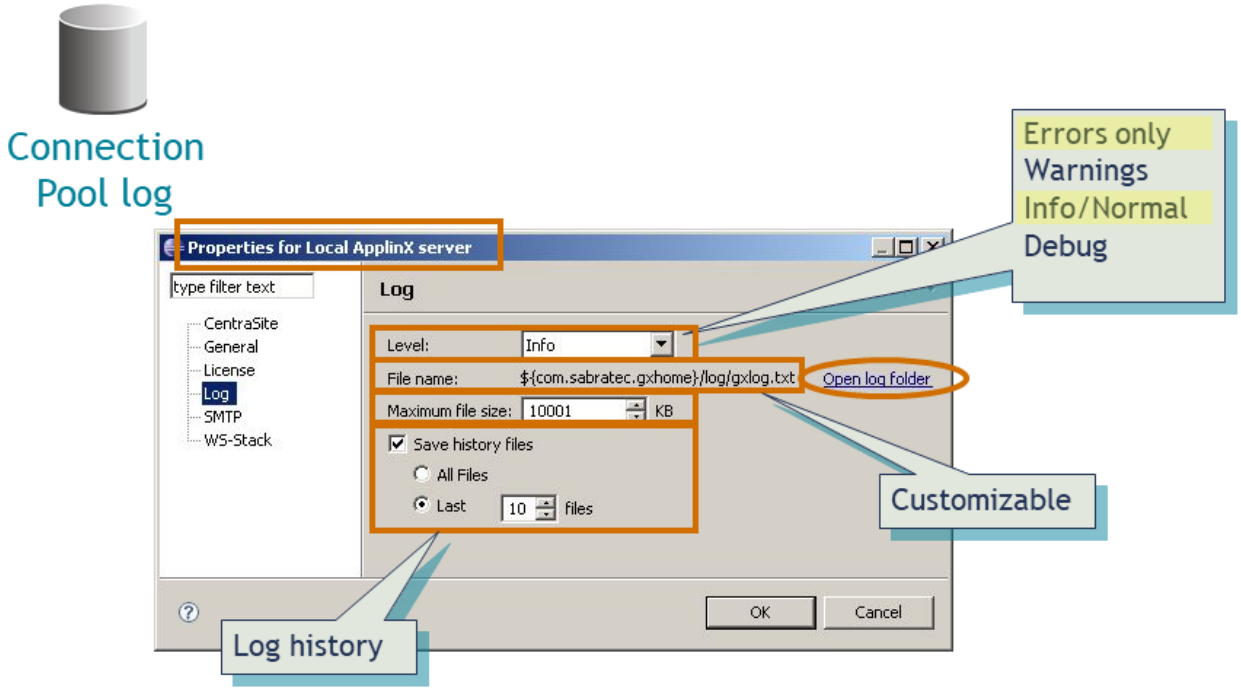
The next portion of lines shows the Connection Pool Log of a Connection Pool. The Connection Pool Log level shown here is "Debug", so we see detailed information about the startup phase of the Connection Pool (e.g. execution of the initialization path).

gxlog.txt

```
INFO  > ApplinX Server 8.0.1.0000.0209 started
INFO  > System info: JVM Vendor=[Sun Microsystems Inc.] Version=[1.5.0_15]
INFO  > ApplinX Server started
WARN  > Default Application not specified. Using SOADemo as default
INFO  > Listening to port 2323 started
INFO  > ************************************************
INFO  > * ApplinX Server 8.0.1.0000.0209 is running             *
INFO  > * Copyright(c) Software AG Ltd.  All rights reserved. *
INFO  > ************************************************
INFO  > Total Memory: 6636KB, Free Memory: 2763KB, Maximum Memory: 379MB
INFO  > INSURANCE:U0000001: Session connection ID: 1
INFO  > INSURANCE:U0000001: Connected. Address: 127.0.0.1
WARN  > <SYSTEM>:Administrator: Wait condition timed out
ERROR > <SYSTEM>:Administrator: Error in procedure [/connect failing].
INFO  > Service [ONLINE,2] INITIALIZER: Prepare initializer with path
connect_logon
INFO  > Service [ONLINE,2] INITIALIZER: Set dynamic config with info id: 1
INFO  > Service [ONLINE,2] INITIALIZER: Start connection
INFO  > Service [ONLINE,2] INITIALIZER: Waited for specific next screen, rc = 0
INFO  > Service [ONLINE,2] INITIALIZER: Run initialization path
INFO  > Service [ONLINE,2] PATH RUNNER: Setting application parameter UserId
INFO  > Service [ONLINE,2] PATH RUNNER: Setting application parameter Password
INFO  > Service [ONLINE,2] PATH RUNNER: Starting path connect_logon
INFO  > Service [ONLINE,2] PATH RUNNER: path returned. rc = 0
INFO  > Service [ONLINE,2] PATH RUNNER: path returned. rc = 0
INFO  > Service [ONLINE,2] INITIALIZER: Adding connection to pool
```

Connection Pool Log

The Connection Pool Log adds Connection Pool specific information to the Server Log file (*gxlog.txt* by default). It is used for fine-tuning Connection Pool parameters or identifying problems of the different Connection Pools.



Configuring the Connection Pool Log

To configure the Connection Pool Log using the ApplinX Administrator

1 In the ApplinX Administrator, under **Management > Current Activity > Connection Pools**, right-click on the Connection Pool for which the Connection Pool Log is to be enabled.

2 Choose **Open** and find the log level on the **General** tab.

To configure the Connection Pool Log using the ApplinX Designer

1 In the ApplinX Designer, open the Connection Pool entity.

2 On the **General** tab, you can set the Log level of the Connection Pool Log.

You can set a different detail of logging for each Connection Pool: **None**, **Errors**, **Warnings**, **Information** and **Details**. Once the application is in production, the level **Error** (which is the default) is recommended. The Server log should be set to no less than level **Info/Normal** in order to see Connection Pool Logs.

Note: No restart is required if you change the Connection Pool Log configuration.

See **example above** and also *Viewing Server Logs* in the Administration documentation.

Process Tracing

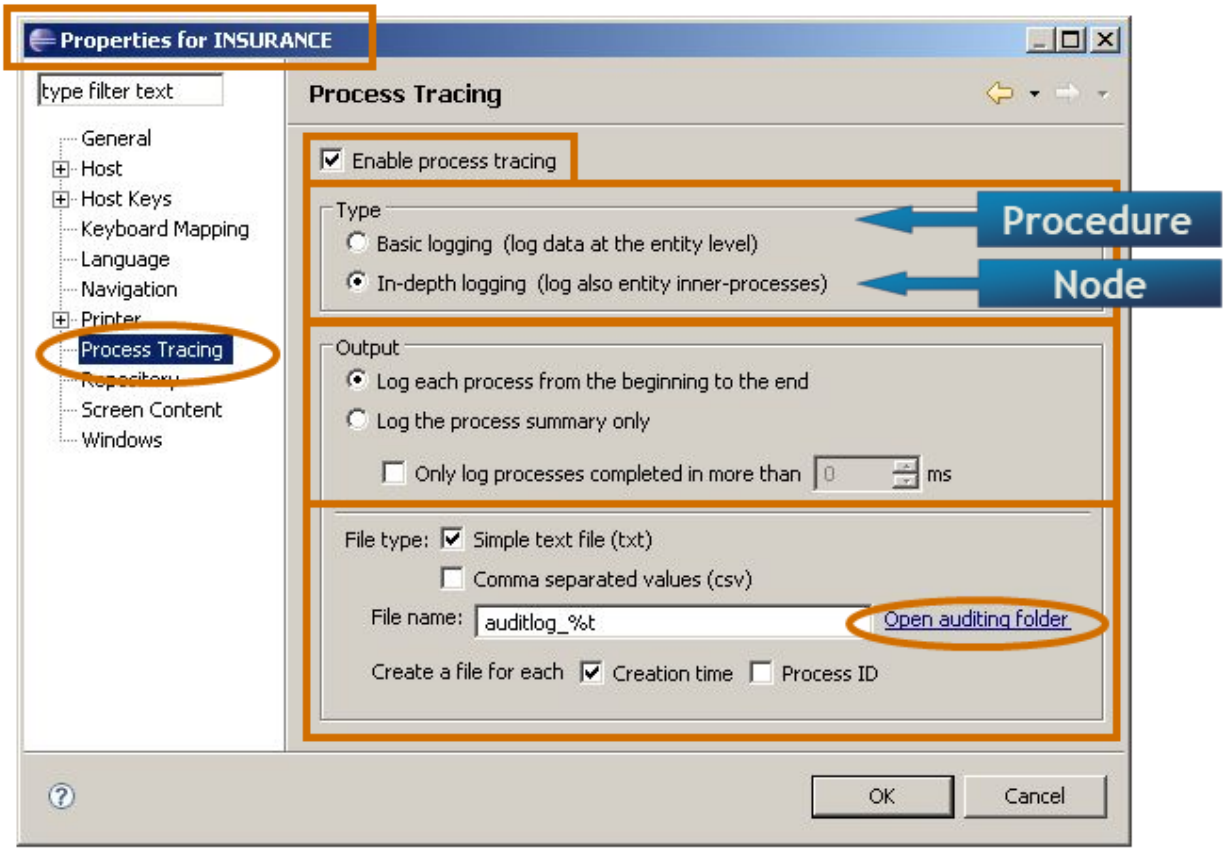
Application Process Tracing is used to provide a log of performance times of processes such as procedures, paths and programs. It is particularly helpful in SOA-enablement applications. This can give a more specific indication (pinpoint the exact process) as to the cause of application performance problems. The application process tracing information can be saved as a .txt and/or .csv file.

Note: The host times that the tracing application process displays are only relevant for hosts that are not character mode hosts (e.g. not VT).

- Configuring Process Tracing

- Process Tracing Example

Configuring Process Tracing



To configure Process Tracing

- In the ApplinX Designer, get to the application's properties and select the **Process Tracing** node. You can change the following settings:

Setting	Description
Enable process tracing	Check this box to activate process tracing in your application.
Type	Select the type of tracing you require: either basic logging, which logs the data at the entity level (logs when a procedure, path or program started and when they were completed) or in depth logging, which logs more detailed data, at the step or node level.
Output	Select whether to log each process from the beginning to the end or just to log a summary of the process. When logging just the process summary, it is possible

Setting	Description
	to define to only log processes that took longer than a certain amount of time to be completed.
File type	Indicates whether to display the logged data in a simple text file (.txt) and/or as comma-separated values (.csv) format.
File name	The name of the trace file. You can create a separate File for each session, or connection and the name can include the creation time and/or process ID. `%t` will insert the time stamp of the connection. `%p` will insert the process ID (connection or session ID).
Open auditing folder	Opens the Windows Explorer and displays the location and list of existing files.

Note: No restart is required if you change the Process Tracing configuration.

Process Tracing Example

This example of Process Tracing shows the trace from running a flow procedure named "getCustomers_v3". The outcome is, that executing this procedure took 765 ms in total. As the log level used was in-depth logging, the trace also contains execution times for the single nodes inside the Flow Procedure. One of these nodes is about executing a Path Procedure, which takes 688 ms … this is close to 90% of the total execution time of the procedure.

auditlog_xxxxx.txt

```
INFO > FLOW      STRT Flow INSURANCE:/getCustomers_v3 started
INFO > NODE      STRT   Node INSURANCE:/getCustomers_v3.Emulation Session_1 started
INFO > HOST-CON FIN  Host Connection INSURANCE:Host connection finished. Total time = 49312. [U
INFO > HOST-CON STRT Host Connection INSURANCE:Host connection started. [User: U0000002]
INFO > NODE      FIN    Node INSURANCE:/getCustomers_v3.Emulation Session_1 finished. Total time
INFO > NODE      STRT   Node INSURANCE:/getCustomers_v3.TryCatch_1 started
INFO > NODE      FIN    Node INSURANCE:/getCustomers_v3.TryCatch_1 finished. Total time = 0
INFO > NODE      STRT   Node INSURANCE:/getCustomers_v3.TryCatch_1_Try started
INFO > NODE      STRT     Node INSURANCE:/getCustomers_v3.Exec_Proc_1 started
INFO > NODE      FIN      Node INSURANCE:/getCustomers_v3.Exec_Proc_1 finished. Total time = 688
INFO > NODE      STRT     Node INSURANCE:/getCustomers_v3.Mapper_1 started
INFO > NODE      FIN      Node INSURANCE:/getCustomers_v3.Mapper_1 finished. Total time = 0
INFO > NODE      STRT     Node INSURANCE:/getCustomers_v3.Mapper_2 started
INFO > NODE      FIN      Node INSURANCE:/getCustomers_v3.Mapper_2 finished. Total time = 0
INFO > NODE      STRT     Node INSURANCE:/getCustomers_v3.EndSession_1 started
INFO > HOST-CON FIN  Host Connection INSURANCE:Host connection finished. Total time = 704. [Use
INFO > HOST-CON STRT Host Connection INSURANCE:Host connection started. [User: <No session>]
INFO > HOST-CON FIN  Host Connection INSURANCE:Host connection finished. Total time = 0. [User:
INFO > NODE      FIN      Node INSURANCE:/getCustomers_v3.EndSession_1 finished. Total time = 31
INFO > NODE      FIN    Node INSURANCE:/getCustomers_v3.TryCatch_1_Try finished. Total time = 71
INFO > FLOW      FIN  Flow INSURANCE:/getCustomers_v3 finished. Total time = 765
```

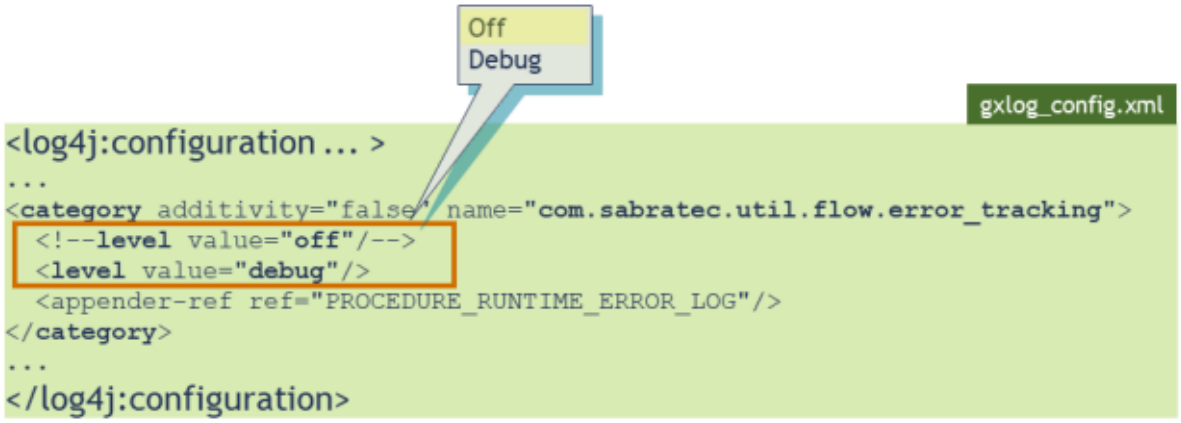
Path Procedure Failure Log

The Path Procedure Failure log includes debug data regarding procedures that fail in runtime. The log includes a snapshot in ASCII characters of the screen that caused the failure.

Configuring a Path Procedure Failure Log

To configure a Path Procedure Failure Log

1 In the Log Configuration file (*gxlog_config.xml*), search for the `<category>` tag with name "`com.sabratec.util.flow.error_tracking`", and change the value in the `<level>` tag from "`off`" to "`debug`".



The name of the created log file will include identifying information about the user and the failing procedure, and a timestamp.

2 Restart the ApplinX server for these changes to take effect.
By default the log is created in the *<ApplinX home>/host-applications/<APPLICATION_NAME>/log* directory, and the file names will have the following format: `debugging_error_in_%I_%t.log`,

where `%I` identifies the user and procedure name, and

`%t` is the timestamp.

The location and name of the file can be changed in the *gxlog_config.xml* file.

Path Procedure Failure Example

Here is an example of a Path Procedure Failure log showing the following:

- message that the execution of a procedure named "connect_failing" failed
- a stack trace
- snapshot of the current screen when the failure occurred

debugging_error_in_xxxxx.log

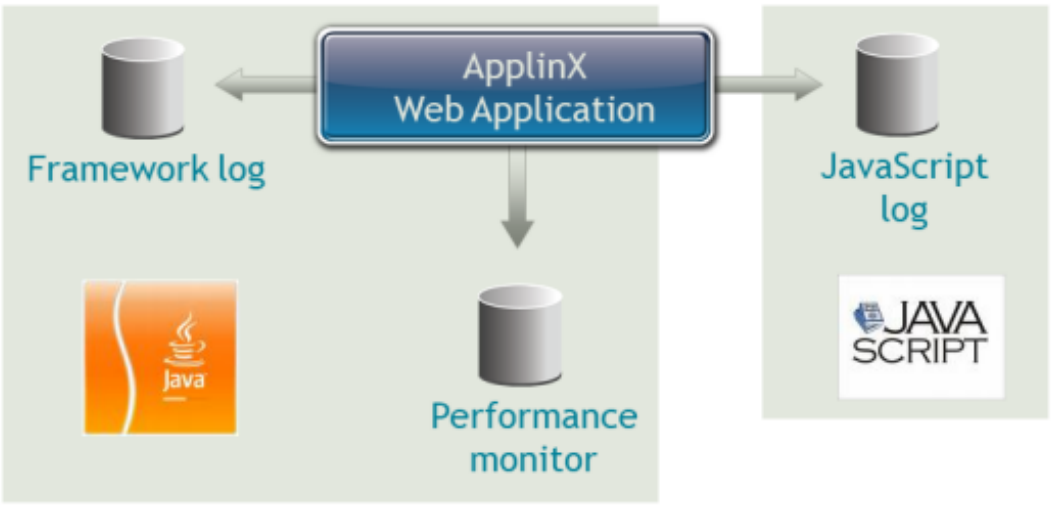
```
19/02/2009 12:29:17.953> An error has occurred during a procedure call [/connect_failing]. Procedure debug data:
Flow exception occured on [Exception 1]
Message: Didn't get to Screen D_Logon
Original Exception: java.lang.Exception: Didn't get to Screen D_Logon
--- STACK TRACE BEGIN ---
            at com.sabratec.util.flow.GXThrowExceptionNode.doProcess(Unknown Source)
            at com.sabratec.util.flow.GXAbstractExpressionContainerNode.processInner(Unknown Source)
               ...
--- STACK TRACE END ---
19/02/2009 12:29:17.968> Last screen [/D_Start] content:

  1234567890123456789012345678901234567890123456789012345678901234567890123456789
 1   TERMINAL : DAEHTC27
 2                         SSSSSSSSSSSSSSS        SOFTWARE AG
 3                      SSSSSSSSSSSSSSSSSS
 4                    SSSSSSSSSSSSSSSSSSSS     Education/Training
 5                   SSSSSSSS
 6                  SSSSSSSS
 7                  SSSSSSSS       SSSSSSSS          HH    HH
 8                  SSSSSSSS       SSSSSSSS          HH    HH
 9                  SSSSSSSS       SSSSSSSS          HH    HH
10                  SSSSSSSS       SSSSSSSS          HHHHHHHH
11                  SSSSSSSS       SSSSSSSS          HH    HH
12                                 SSSSSSSS          HH    HH
13                                SSSSSSSS           HH    HH
14                  SSSSSSSSSSSSSSSSSSS
15                  SSSSSSSSSSSSSSSSSS
16                  SSSSSSSSSSSSSS                Environment
17
18  For Training Courses, please enter 'C'
19  For System Admistration Courses, please enter 'A'
20
21 Desired Target:
22
23  IP-addr: 10.20.122.63   :02928
24  Host:
```

Log Files on the ApplinX Web Application Side

This section covers the following topics:

- JavaScript Log

Scope

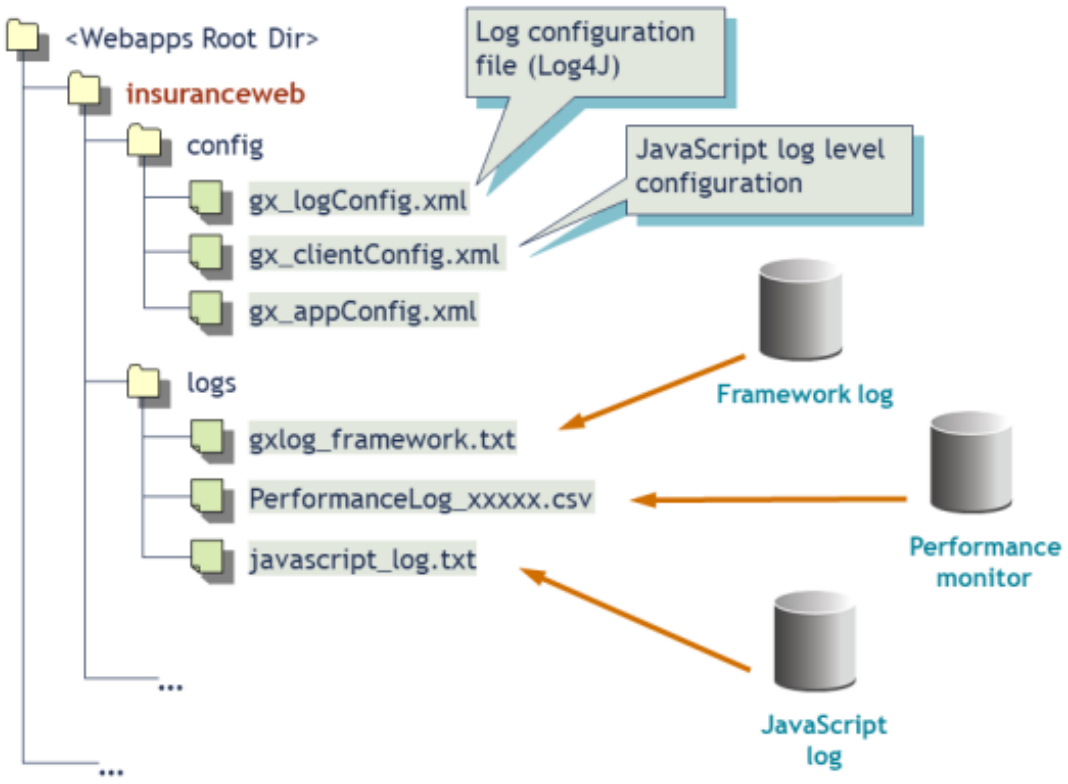


There are three log files on the web application side:

- **Framework Log**
  This is the main log file of an ApplinX web application. It includes general log data for all applications available on the server (e.g. server startup and shutdown, application loading, session start and end). Its log level can be configured on the level of the ApplinX Server.
- **Performance Monitor**
  This can be enabled to measure execution times and to analyze performance problems.
- **JavaScript Log**
  This log is used primarily to log JavaScript errors and also to debug specific modules of the JavaScript engine.

Log File Locations

The following graphic shows the location of the log configuration files and the individual log files.



The following configuration files are standard Log4J configuration files located in the *config* subfolder inside the web application's root directory on the application server:

- **Web Log Configuration File**
  File *gx_logConfig.xml* contains all of the log configuration of the web application (with the exception of JavaScript).
- **JavaScript Log Configuration File**
  File *gx_clientConfig.xml* contains the configuration of the JavaScript log level.

The following log files are created by default in the *logs* subfolder of the web application's root directory:

- **Framework Log**
  This file, by default, is written to a file named *gxlog_framework.txt*.
- **Performance Monitor**
  The Performance Monitor, when enabled, is written to a file named *PerformanceLog_xxxxx.csv* where the extension can also be .txt. The file name may include the session ID.
- **JavaScript Log**
  This log is written, by default, to the file *javascript_log.txt*.

Names and locations of all three log files can be customized in the log configuration file *gx_logConfig.xml*.

Framework Log

Configuring the Framework Log

The Framework log is the principal log file of the ApplinX Web Application. It supports multiple log levels and log history.

ApplinX provides a web-based Configuration Editor where you can configure framework parameters, including those for logging. See Web Application Configuration Parameters in the Web Application Development documentation.

You can use the ApplinX Framework log to analyze and debug the code in the Web application. This is implemented by making changes in the *config/gx_logConfig.xml* file. See *Investigating the Web Application's Code* in the Web Application Development for further details.

To configure the Framework log

1 Open a new browser and run your Web application.

2 On the ApplinX Framework starting page, click on the **Configuration** link. The Configuration Editor will be displayed.

3 Expand the node named **Log** and specify the following:

Setting	Description
File name	The log file name and path relative to the web application's root directory.
Append to existing file	Check this box to append to an existing log file instead of starting a new one.
Log level	Multiple levels, from "Errors only" to "Debug" are supported. The "Normal" level is default and "Errors only" is recommended for an application in production.
Log history	Specify the maximum number of old log files to keep; when this number is reached, the oldest log history file is deleted when the next log history is created.
Max. file size	The maximum file size (in bytes) for a log file; when the log file reaches this limit, it is saved to a log history file, and a new Framework log is started.

4 Remember to save your changes in the Configuration Editor.

Note: No restart is required if you change the Framework log configuration.

Framework Log Example

The following is an example of a Framework log. The log level in this case was set to "Debug", and the highlighted area shows the lifecycle trace created by a page named "D_Logon.jsp" when this was loaded.

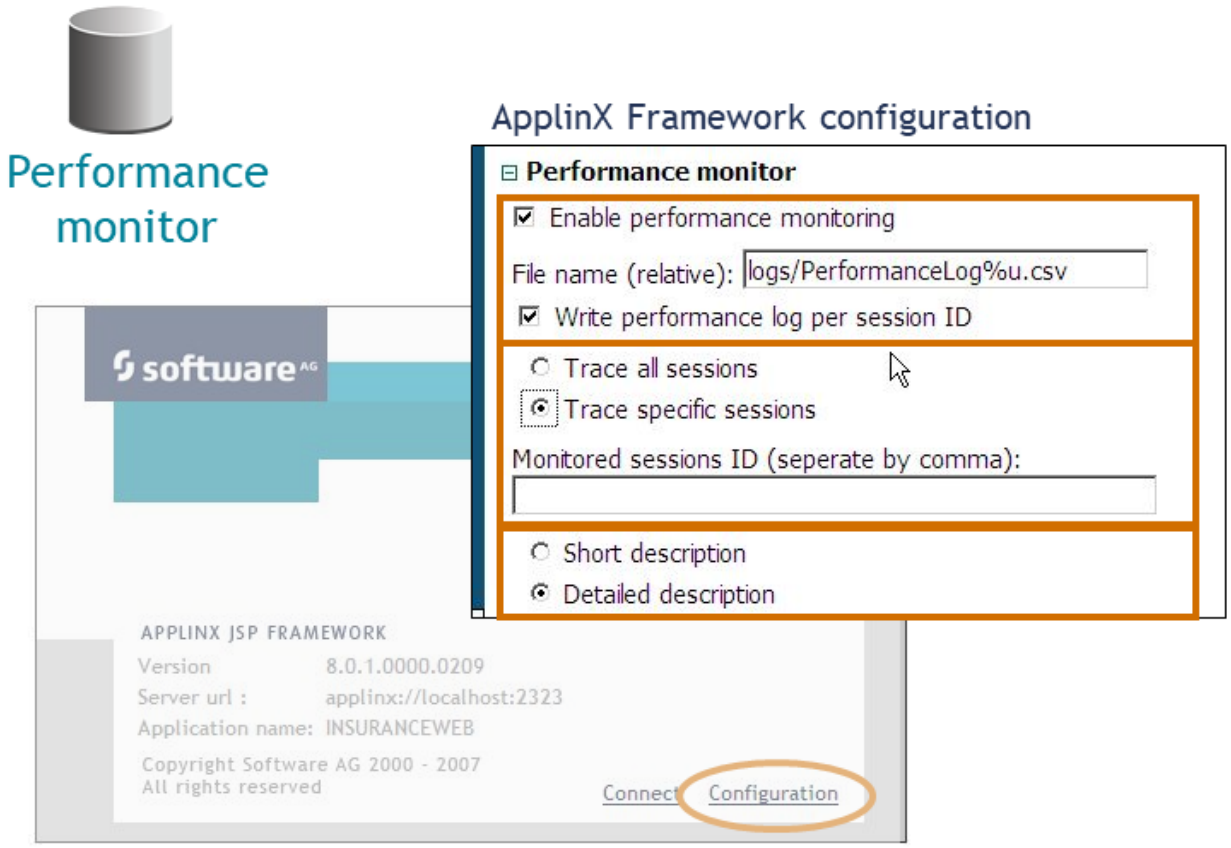
gxlog_framework.txt

```
DEBUG Initalizes some session variables with null
WARN  redirect to URL D_Logon.jsp has been canceled due to a previous redirect call
DEBUG xxxxx-gxfirstpage.jsp, Loading actions using no cache
DEBUG response was commited after gx_onInit stage, performing unload
DEBUG xxxxx-gxfirstpage.jsp, Detaches the base object from ApplinX server
DEBUG xxxxx-gxfirstpage.jsp, keeping baseobject open for next redirected page.
DEBUG xxxxx- Page:http://localhost:18080/insuranceweb/gxfirstpage.jsp ended, base object sessio
DEBUG Initalizes some session variables with null
DEBUG notifyAll on unload-1235130646765
DEBUG synchronizing unload-true-1235130646781
DEBUG xxxxx-D_Logon.jsp, Loading actions using no cache
DEBUG xxxxx-D_Logon.jsp, Performing framework default logic: attach
DEBUG xxxxx-D_Logon.jsp, Session is attached from a prior page.
DEBUG xxxxx-D_Logon.jsp, Throws post connect event
DEBUG xxxxx-D_Logon.jsp, Checks if window resizing is required.
DEBUG xxxxx-D_Logon.jsp, Performing framework default logic: sync
DEBUG xxxxx-D_Logon.jsp, Adding full javascript support to the page
DEBUG xxxxx-D_Logon.jsp, Performing framework default logic: not in post back mode - filling fo
DEBUG xxxxx-D_Logon.jsp, Throws pre fillForm event
DEBUG xxxxx-D_Logon.jsp, Fills page tags/controls
DEBUG xxxxx-D_Logon.jsp, Applies protected, trim & colors definition to table configuration
DEBUG xxxxx-D_Logon.jsp, Detaches the base object from ApplinX server
DEBUG xxxxx-before detach request to server
DEBUG xxxxx-after detach request to server
DEBUG xxxxx-D_Logon.jsp, after framework detaching.
DEBUG xxxxx- Page:http://localhost:18080/insuranceweb/D_Logon.jsp ended, base object session sc
```

Performance Monitor

Configuring the Performance Monitor

The Performance Monitor can be enabled to measure execution times in an ApplinX web application. This can be helpful to analyze performance problems, for instance to find out whether time is consumed mostly in the web application or in the ApplinX Server.



To enable and configure the Performance Monitor

1 Open the web-based Configuration Editor.

2 Expand the node named **Performance monitor** and specify the following:

Setting	Description
Enable performance monitoring	Check this box to enable the Performance Monitor.
File name	The log file name and path relative to the web application's root directory.
Write performance log per session ID	Check this box to get a separate log file per session ID (%u is added to the file name).
Trace all/specific sessions	Specify here whether to take all sessions or only selected sessions into account.
Monitored sessions ID	If **Trace specific sessions** was selected, provide here a comma-separated list of the session IDs to monitor.
Short/detailed description	Select here the level of detail in the log.

3 Remember to save your changes in the Configuration Editor.

Note: No restart is required if you change the Performance Monitor configuration.

Performance Monitor Example

This example of a Performance Monitor log shows, for example, that loading of the page I_Menu.jsp took 1141 ms in total, 1015 ms of which were consumed at the ApplinX Server side (which is about 89% of the time).

It also shows additional times depending on the type of page (Instant and Generated page).

PerformanceLog_xxxxx.csv

```
Screen name,D_Logon,Generated page
Total request,351906
Total Server side,351609
Attach,250,SendKeys,406,Fill form fields,0,Fill table,0,Detach,0
Client Load,203,C92969BC8F21A3E200BCD5B4FD870EE8

Screen name,I_Menu,Instant page
Total request,1141
Total Server side,1015
Attach,15,SendKeys,0,Generate Instant,0,Detach,0
Client Load,31,C92969BC8F21A3E200BCD5B4FD870EE8

Screen name,I_Customers1,Generated page
Total request,1313
Total Server side,1172
Attach,563,SendKeys,343,Fill form fields,0,Fill table,0,Detach,0
Client Load,63,C92969BC8F21A3E200BCD5B4FD870EE8

Screen name,I_Proposals,Instant page
Total request,1266
Total Server side,969
Attach,437,SendKeys,0,Generate Instant,0,Detach,0
Client Load,141,C92969BC8F21A3E200BCD5B4FD870EE8

Screen name,I_ProposalDetails1,Instant page
Total request,563
Total Server side,437
Attach,15,SendKeys,422,Generate Instant,0,Detach,0
Client Load,47,C92969BC8F21A3E200BCD5B4FD870EE8
```

JavaScript Log

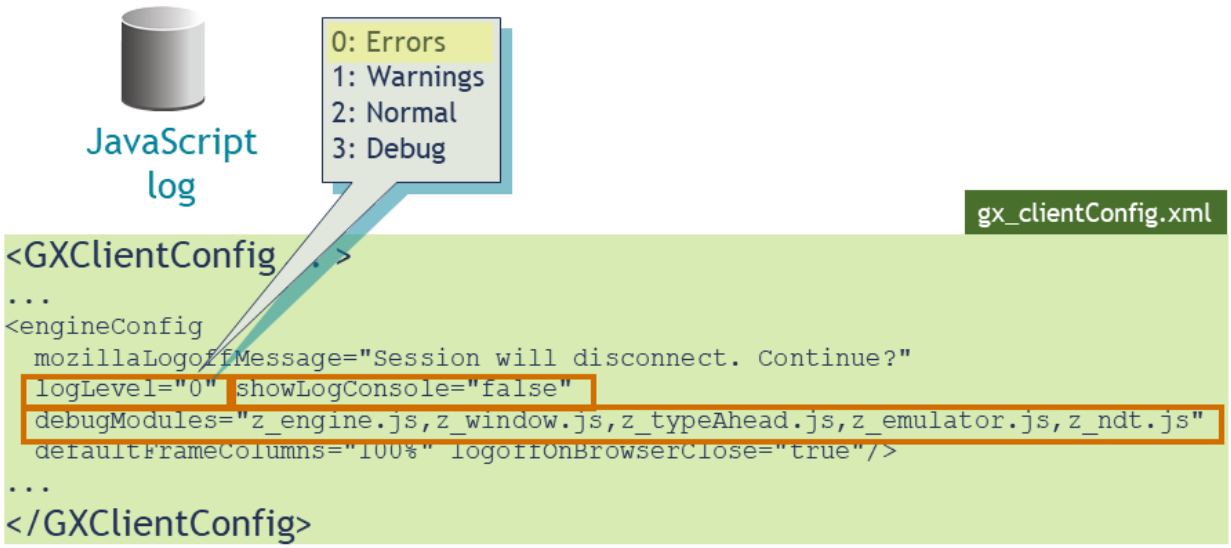
- Configuring the JavaScript Log

- JavaScript Log Example

Configuring the JavaScript Log

The JavaScript log is used to debug specific modules of the JavaScript engine. The engine logs JavaScript errors that occur to users of the web application. These errors are logged to the framework JavaScript log.

The JavaScript logger engine automatically logs JavaScript errors to the JavaScript log file. See JavaScript Logger Engine in the *Web Application Development* for more information.



To configure the JavaScript log

1 Open the configuration file *gx_clientConfig.xml* in an editor and search for the `<engineConfig>` tag.

2 In the `logLevel` attribute, set the log level:
0: Errors (default)
1: Warnings
2: Normal
3: Debug

3 If the `showLogConsole` attribute is set to `true`, the log output is also displayed in a log console in the browser.

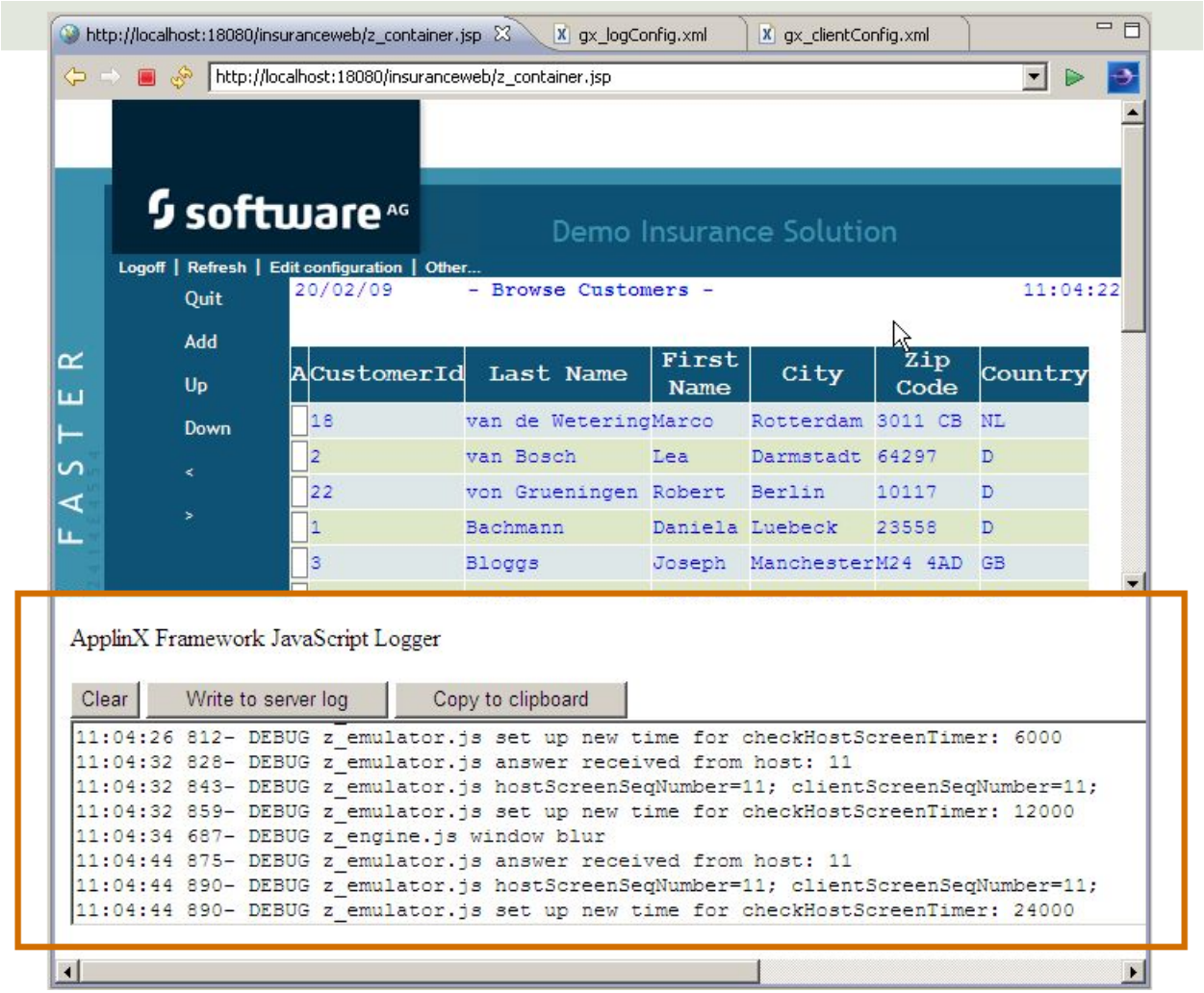
4 The `debugModules` attribute specifies, as comma-separated values, the list of modules of the JavaScript engine to log. These modules are:

- JavaScript engine core (`z_engine.js`)
- Modal Windows (`z_window.js`)

- Type ahead (`z_typeAhead.js`)
- Emulation (`z_emulator.js`)
- Natural Data Transfer (`z_ndt.js`)

Note: No restart is required if you change the JavaScript Log configuration.

This example shows how the JavaScript Log Console, when enabled, is displayed inside the browser window:



JavaScript Log Example

This is an example of a JavaScript log created from the page D_Logon.jsp.

The fragment in the middle shows the tracing of the action of pressing the Enter key, which is mapped to a click on the **Logon** button.

javascript_log.txt

```
DEBUG z_engine.js JavaScript engine loaded for page http://localhost:18080/insuranceweb/D_Logon
DEBUG z_emulator.js Setting caret position to 0
DEBUG z_emulator.js Field content UserId is selected. Select field on focus is configured
DEBUG z_emulator.js answer received from host: 5
DEBUG z_emulator.js hostScreenSeqNumber=5; clientScreenSeqNumber=5;
DEBUG z_emulator.js set up new time for checkHostScreenTimer: 6000
DEBUG z_emulator.js Setting caret position to 0
DEBUG z_emulator.js Field content Password is selected. Select field on focus is configured
DEBUG z_engine.js Executing keyboard mapping:
function login() {
  document.getElementById('logonBtn').click();
}
DEBUG z_engine.js Page is submitted with the following content:
UserId:da001
Password:***
POS1223:***
POS1303:
POS1543:X
DEBUG z_emulator.js Detaching all input tag events
DEBUG z_emulator.js config.attachInputTagEventsfalse
DEBUG z_emulator.js ActiveX component is not enabled
DEBUG z_emulator.js reset timer due to postBack event
DEBUG z_engine.js in lock screen, GXScreenLocker.__isEnabled=true
DEBUG z_engine.js in lock screen, scrLock==null:false
DEBUG z_engine.js in lock screen, scrLock.style.zIndex:-1
DEBUG z_emulator.js Setting focus to clicked taglogonBtn
DEBUG z_emulator.js Updating cursor hidden to field logonBtn
DEBUG z_engine.js frame unload
```

Other Log/Trace Files

The following logging and tracing possiblities are also provided:

- GCT Trace File

- Host Print Log

GCT Trace File

The GCT Trace File log enables recording a file, which traces the connection communication (connection pool or user) between the ApplinX server and the host, for each connection. It is possible to define whether a single trace file will be created, replacing the previously saved file or whether the data will be saved to a new file on every new connection or session. Identifying the separately saved files is possible by inserting identifying parameters in the file name (the session ID, creation time and/or connection ID). It is highly recommended to create a separate file for each session/connection or creation time. Note that trace files can be created from within the **session definition** overriding the application definition. This is recommended as it does not conflict with other existing sessions.

Refer to **Recording Trace Files** for more information.

Refer to the **Session View** for details regarding overriding application properties per session and also for details on navigating within a session in replay mode.

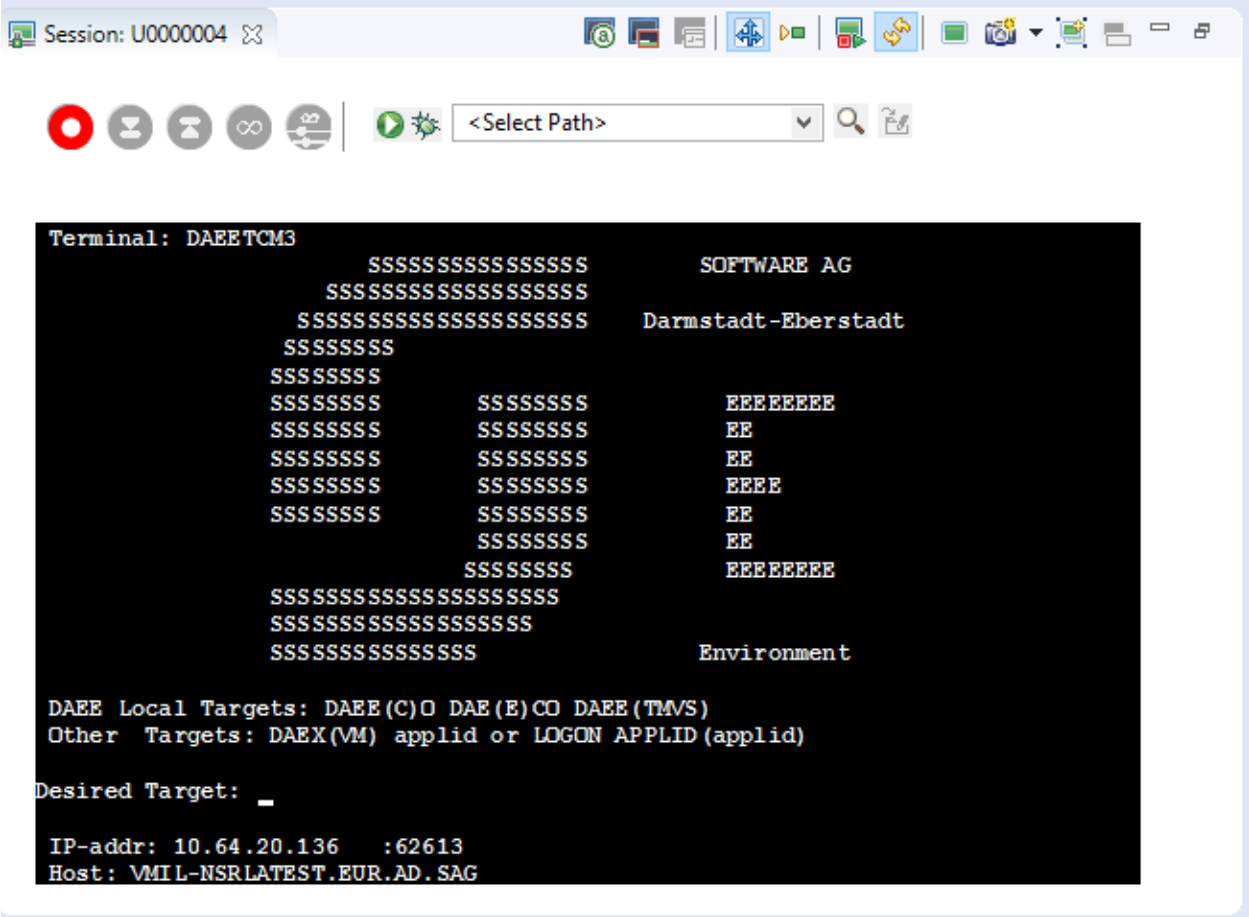
Host Print Log

The Host Print Log logs information regarding the printing activity. This information can be displayed by clicking on the **Show Log** button. There are two log levels which can be set: normal (default) or debug. The `loglevel` parameter determines the log level.

II Sessions

9 The Session View

The Session View is especially useful when working with screens or path procedures, either offline (replay file) or online.



For further details regarding the Session View Toolbars refer to Session View reference.

Creating a Display Session

The definitions configured here override application definitions and are relevant for a specific session.

To create a display session

1 Open the relevant application, right-click on the **Sessions** node and select **New Display Session Definition**.

2 Enter a name and description for the session.

3 Select the connectivity type:

- **Use application configuration**: Use the configuration set in the Application Properties.
- **Online**: Connect online to the host, and enter the name of the device.
- **Offline (using trace files)**: Connect to the session using a trace file. Browse and select the required trace file.
- **Connection Pool**: Select a connection pool from which to take the session.

4 Click **Next** to define the Session General Properties.

5 Enter the session ID and password.

6 Select whether or not to create a trace file and where and how it should be saved.

7 Click **Finish**.

Duplicating a Session

Duplicating a session will create a new session definition will be created using the same configuration as the original one. Session definitions can be duplicated by right-clicking on the original session and selecting **Duplicate Session Definition**.

Opening an HTTP Session

Opening an HTTP session opens the session as an Instant Web Application. The session is displayed by default in the Editor area of Eclipse. It is possible to display the session in your system's browser by changing the Web browser definitions in Eclipse (**Windows>Preferences>General>Web Browser**). This is convenient for checking transformations. This is not available when you do not have a Web Enablement license.

To open an HTTP session

- Right-click on the relevant session within the session node of the application or access the **Session** menu and select **Open HTTP Session**.

Changing the Screen Definition Mode

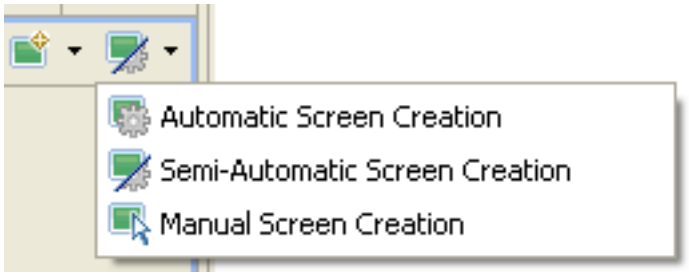
Screen Creation Definitions are a set of basic definitions which are used to create a new ApplinX screen. These definitions include the default screen name, initial identifiers and determine whether input fields will be created. Creating a new screen using screen creation definitions reduces the complexity and the time required to identify new screens.

How does it work?

To use this feature, at least one set of Screen Creation Definitions must be created within a Screen Group (in the dedicated tab). When navigating within a session, and reaching an unidentified screen, ApplinX searches for Screen Groups which match the unidentified screens, and then creates a new screen based on the screen group's identifiers and on parameters defined in the creation definition configured in the Screen Group. This process can be initiated in three different modes:

- Automatically: A screen will automatically be created for each unknown screen that matches a screen group. This screen will include basic screen definitions, which are defined in the screen group.
- Semi-automatically: The New Screen Wizard will automatically be displayed for each unknown screen that suits a screen group. This screen will include basic screen definitions, which are defined in the screen group.
- Manually: The New Screen Wizard is displayed when clicking on the Identify New Screen icon.

The desired mode is determined by clicking on the **Change Screen Definition Mode** icon in the Session View.



New Screen/Screen Group

An unidentified screen can be identified as a screen/screen group by clicking on the New Screen icon in the Session Toolbar. The Create New Screen wizard is displayed. Refer to Creating a New Screen or Creating a New Screen Group for further details.

Updating a Screen's or Screen Group's Image

The screen image of a screen can sometimes change and no longer suit the relevant screen entity. In the Session View there is a **Update Screen Image** icon, which enables updating the screen image. Click on the arrow next to the icon to view all the relevant screens/screen groups and select the desired one.

Recording a Path

A Path Procedure can be created either via the Session view or by creating a New Path Procedure entity (Refer to Creating a Path Procedure for further details). When creating the Path procedure in the Session, use the Path toolbar to record the path and then you can edit the path in the Path procedure dialog box.

To create a Path procedure from the Session

1 Navigate to the first screen from which you want the recording to commence.

2 Click on the Path toolbar icon to display the Path toolbar.

3 Click on the Record icon.

4 Navigate to the various screens to record the relevant path.

5 Click on the Define Inputs icon to mark all the unprotected input fields in purple. Click on one of the marked fields to define it as an input. The field will be marked orange.

6 Click on Define Procedure Outputs to mark all the protected and unprotected fields. Click on a field to define it as an output in the Path procedure. The field will be marked blue.

7 If you want to record a step that repeats again and again until a certain condition is reached, click the loop icon, and follow the instructions in the wizard. This kind of definition can be applicable for collecting data of a host table.

8 To end a recording click the Stop icon. The New Path Procedure wizard is displayed where you are required to enter a name for the path procedure. Once you click Finish, the Path Procedure is opened in the Editor area where you can add/change the procedure as required.

Running/Debugging a Path Procedure

A Path Procedure can be run/debugged in the session viewer, by selecting a path procedure from the list and then clicking on the run/debug icon.

When running a path procedure, you will be required to insert inputs as necessary.

When debugging a path procedure, the Debug perspective will open.

Testing the Application Map

The screen navigation defined in the Application Map can be tested to ensure that the navigation behavior is as expected. This is done in the Session View, using the Application Map toolbar. The toolbar enables selecting a screen to which you expect to be able to navigate to from the current screen and then attempting to navigate to this screen. If the Application Map is correctly defined, then ApplinX will successfully navigate to the selected screen. If the Application Map does not have the relevant steps defined to reach the screen you selected or if these steps are not "Approved" steps, a pop-up message will inform you of this.

Note: The Application Map toolbar is not available when working offline.

Refer to Application Map Toolbar.

Navigating within a Session: Character Mode (VT) Hosts

Character mode (VT) host screens consist of unprotected fields only. Therefore the regular working mode of navigating through screens - typing data in input fields and pressing a key in order to send data to the host, cannot be implemented for VT hosts. There are two alternatives to working with VT applications: working in Character mode or working in Block mode.

- Character mode: Simulates the host behavior where each key stroke is submitted to the host.
- Block mode: Simulates ApplinX behavior, where the whole screen is sent to the host when clicking the Enter key at the end.

In the application development process, Character mode should be used in the early process of development, mainly for screen navigation. In advanced stages of development, Block mode should be used in order to resemble SOA working mode (which is, by definition, Block mode).

The desired mode is determined by clicking on the **Block mode/Character mode** icon in the Session View.

For further information regarding character mode hosts refer to Character Mode Hosts (VT Protocol).

Navigating within a Session

Within a session, you can navigate online, interacting directly with the host, or to replay a previously recorded navigation session. When working directly with the host, you can create a trace file that records the navigation between the screens. This can later be used and replayed, enabling you to work offline (in replay mode).

Online Mode Navigation

Online session navigation using the session view is similar to regular host navigation:

Using Keyboard Keys

- Use the keyboard to enter any alphanumerical input.
- Use the Enter key to send Enter to the host.
- Use the function keys (F1-F12) or the PF buttons toolbar to send corresponding PF1-PF12 keys to the host.
- Use the SHIFT key combined with the function keys to send PF13-PF-24 keys to the host (Shift+PF1=PF13, Shift+PF2=PF14 and so on).
- Use the Customize Host Keys to define and send any other host key.

Creating Trace Files

When working online, you can use the Trace file feature to record a file, and trace the connection communication (connection pool or user) between the ApplinX server and the host, for each connection. Details for setting the default trace file parameters can be found in Recording Trace Files. You can override the application settings and define settings per session either when creating a new session connection or in the session properties of an existing session.

Replay Mode Navigation

In order to debug or reconstruct specific scenarios, you can replay (GCT) files that have traced the steps of these scenarios. In order to replay such files, the application configuration definitions must indicate that instead of working online with the host, ApplinX must, when connecting, access and replay the specific file. The default trace file used is defined in the Application Properties dialog box (refer to Working with Offline Sessions for further details). To override these settings, define the relevant settings per session either when creating a new session connection or in the session properties of an existing session.

When replaying a file, you may want to navigate to a specific screen. The replay navigator enables this, simply and efficiently, using the Replay Navigator slider. Note: In the replay mode, any function keystroke (such as [ENTER], PF keys etc.) displays the following screen, according to the screen order in the recorded GCT file, therefore, key selection is meaningless. This feature is not available when connecting to a host connection pool (online or offline).

In the ApplinX web applications, you can navigate directly to a specific screen in the replay file, according to its serial number in the file: this number can be found from the session properties dialog, or by hovering above the Session View status bar, in the section displaying the name of the replay file. For example, in order to navigate to screen number 53 in the replay file, type ">53" (no quotes) in any input field on the screen. Relative navigation is also possible: going to the previous screen is done by typing ">-1" in any input field.

Using the replay navigator

1. When connecting to a session which has been defined as working offline, in replay mode, (refer to Working with Offline Sessions for further details) the Replay Navigator slider is displayed.

   This toolbar is displayed by default and can be hidden by clicking on the toolbar icon.

   To navigate to the different screens either drag the slider's indicator along until reaching the relevant screen number (the screen number is indicated by a ToolTip), or enter the number in the box to the right of the slider and click the Go arrow or select a screen from the list and click the Go arrow. The slider indicator as well as the number in the box on the right will change according to the present screen number.

   You can search for screens associated with a specific screen group, by selecting the screen group from the list of screens/screen groups. Clicking on Go will search for the first screen encountered, which is associated with the selected screen group. Clicking Go additional times will search for additional screens associated with this screen group.

2. Click on the Show user input icon (the last button) to display the input that the user entered while the GCT was recorded (when the button is not clicked the screen is displayed as it was before the changes were made). When the button is clicked, the contents of the input fields that

were changed appear in red, a string representation of the host key sent appears in the toolbar and the cursor is positioned in the position it was in when the screen was sent to the host.

Note: When replaying a file that has a Connection Pool, the Replay Navigator slider is not available for the Natural UNIX protocol.

Using the Application Map Toolbar

Select the destination screen from the drop down list of screens, then click on the Navigate to screen icon. Refer to Application Map Toolbar.

Note: The Application Map toolbar is not displayed when working offline.

Switching between Online and Replay Mode

Switch between these modes by changing the connection type in the Session Properties, or by editing the file _inc_applconf.asp.

Displaying Windows when Navigating within a Session

When there is a window within a screen, you can define that the screen area outside the window will be grayed out or displayed in regular colors. This is done by clicking on the Window icon on the session toolbar  .

Note: It is not possible to navigate outside the window.

10 Printing - Printer Session

Note: Applet technology is deprecated. An alternative for host printing is provided using Java Web Start technology. See **To test HostPrint in the Web application** under *Getting Started with the ApplinX Printer*.

Testing the Printlet Connectivity to the Host (via the ApplinX Designer)

The object of this task is to verify that ApplinX is able to connect to and activate a printer session. As this is only performed in order to confirm connectivity, the task describes the necessary steps performed using the ApplinX Designer. To deploy ApplinX using the printing options, refer to **Deploying the Printlet.**

To test the Printlet Connectivity:

1 Open the application properties in ApplinX Designer, **Application Properties>Printer** tab, and select the **Enable printer** checkbox. By default, when creating a new application, the printer is enabled.

2 Right-click on the Session node and select New Printer Session Definition.

3 Enter a name for the session definitions.

4 Select the connectivity type:

- **Use application configuration**: Use the configuration set in the Application Properties.
- **Online**: Connect online to the host, and enter the name of the device.
- **Offline (using trace files)**: Connect to the session using a trace file. Browse and select the required trace file.

5 Click Next to display the General Session Properties.

6 When necessary fill in the session ID and password.

7 Determine whether you would like to create a trace file of the session.

8 Click **Finish**. The Printer session definition will be added to the Sessions node of the relevant application.

9 Double-click on the Printer session definitions to test the connection to the host. A new window will come up, containing the ApplinX printlet (printer applet), displaying the printing configuration. The printer status is "Ready". When a print job is sent to the defined device name, the Printer dialog box is displayed and you can print the print job to the local printer.

Deploying the Printlet

When deploying the Web application, the Printlet will need to be configured for a number of users. Each user may require different Printlet behavior for printing jobs. These differences may be for example, using a different device name per user, or using different fonts and layout parameters. Parameters, such as the printer device name that changes according to the current user, must use dynamic code. The dynamic code must retrieve the information (that may be held, for example, in a database or XML file), per user.

Note: The method in which printing jobs are executed may depend on settings such as whether the job will be printed silently without invoking a print dialog box first or whether the print dialog box will be displayed only the first time you print in the current session. These settings are defined in various parameters as detailed in the Printlet API.

To Activate the printer session in the Web application (JSP):

1 Open the page *run_printlet.jsp*.

2 Set the parameters of the printer applet: **application** and **device_name**.

3 Open a browser and run the page *run_printlet.jsp*. The applet is displayed, the status is "Ready". When a print job is sent to the defined device name, the Printer dialog box is displayed and you can print the print job to the local printer.

Notes:

1. The **Options** button is only enabled in Mainframe hosts.
2. The Printlet component is not supported for Microsoft Edge Browser.

To Activate the printer session in the Web application (.NET):

1 Open the page *dotnet\web\cs\New Application\run_printlet.aspx*.

2 Set the properties of the printer control (optionally in the designer): **application**, **device_name**.

3 Open a browser and run the page *run_printlet.aspx*. The applet is displayed, the status is "Ready". When a print job is sent to the defined device name, the Printer dialog box is displayed and you can print the print job to the local printer.

Notes:

1. The **Options** button is only enabled in Mainframe hosts.
2. The Printlet component is not supported for Microsoft Edge Browser.

Launching the Printlet (run_printlet)

There are two options as to how to open the Printlet (run_printlet):

1 Launch the printlet in a new browser window: Add a button to the master page (template.jsp/template.master.cs). Place the button where you desire. Set the button to open the run_printlet.jsp/aspx in a new window.

For example: `<input type="button" value="Print" onclick="window.open('run_printlet.jsp');" />`

Refer to the API:

- `gx_showPrinter`
- `gx_hidePrinter`

Or:

2 Load the printer as an internal frame of your web application (the printlet will always stay active):

1. In the Framework Configuration Editor, in the Emulation node check the **Load printer applet within internal frame** checkbox (refer to Configuring your Web Application) OR In the web application root directory, open the *config/gx_clientConfig.xml* file and set the `LoadPrinter` value to "true".
2. To show the frame of the printer use the `gx_showPrinter()` function.

   To hide the frame of the printer use the `gx_hidePrinter()` function. By default *run_printlet.jsp/aspx* has this function mapped to a button.

   Example: `<input type="button" value="Show Printer" onclick="gx_showPrinter();" />`

Getting Started with the ApplinX Printer

This task will cover the basic steps required in order to achieve a first test print.

To start working with the ApplinX Printer

1 Open the application properties in ApplinX Designer, **Application Configuration>Printer** tab, and select the **Enable printer** checkbox.

2 Click **OK**. A Printer session node will be added to the application node in the Designer.

To test the printer in the ApplinX Designer

1 Click on the **Printer session** node to test the connection to the host. The *Connection* dialog box is displayed. Click Connect to test the connection.

2 The right pane of the Designer displays the printing configuration. The Session status is "Connected to server". When a print job is sent to the defined device name, the Printer dialog box is displayed and you can print the print job to the local printer.

To test hostPrint in the Web application

1 Set the application parameters in file *hostPrint.jsp/aspx*. Add the parameters to the tag `<gx:jnlp>` in `key=value` format. For example:

```
<gx:jnlp  gx_context="contexts.GXInstantLogicContext"  password="" description="" ↩
>
```

Do not modify the `gx_context` parameter.

2 Open a browser and run the page *hostPrint.jsp/aspx*. A JNLP file is downloaded and - depending on the browser - will open automatically or you have to open it manually. Once it is running, the hostPrint window is displayed and the status is "Connected". When a print job is sent to the defined device name, the Printer dialog box is displayed and you can print the print job to the local printer.

Note: The **Options** button is only enabled in Mainframe hosts.

Important: The JNLP file is opened using Java Web Start (javaw.exe), which will no longer be supported after Java 8. If you are using a Java version higher than Java 8 or an open JDK, you will not be able to open the downloaded JNLP file. In this case, use the **OpenWebStart** tool, an open source reimplementation of the Java Web Start technology.

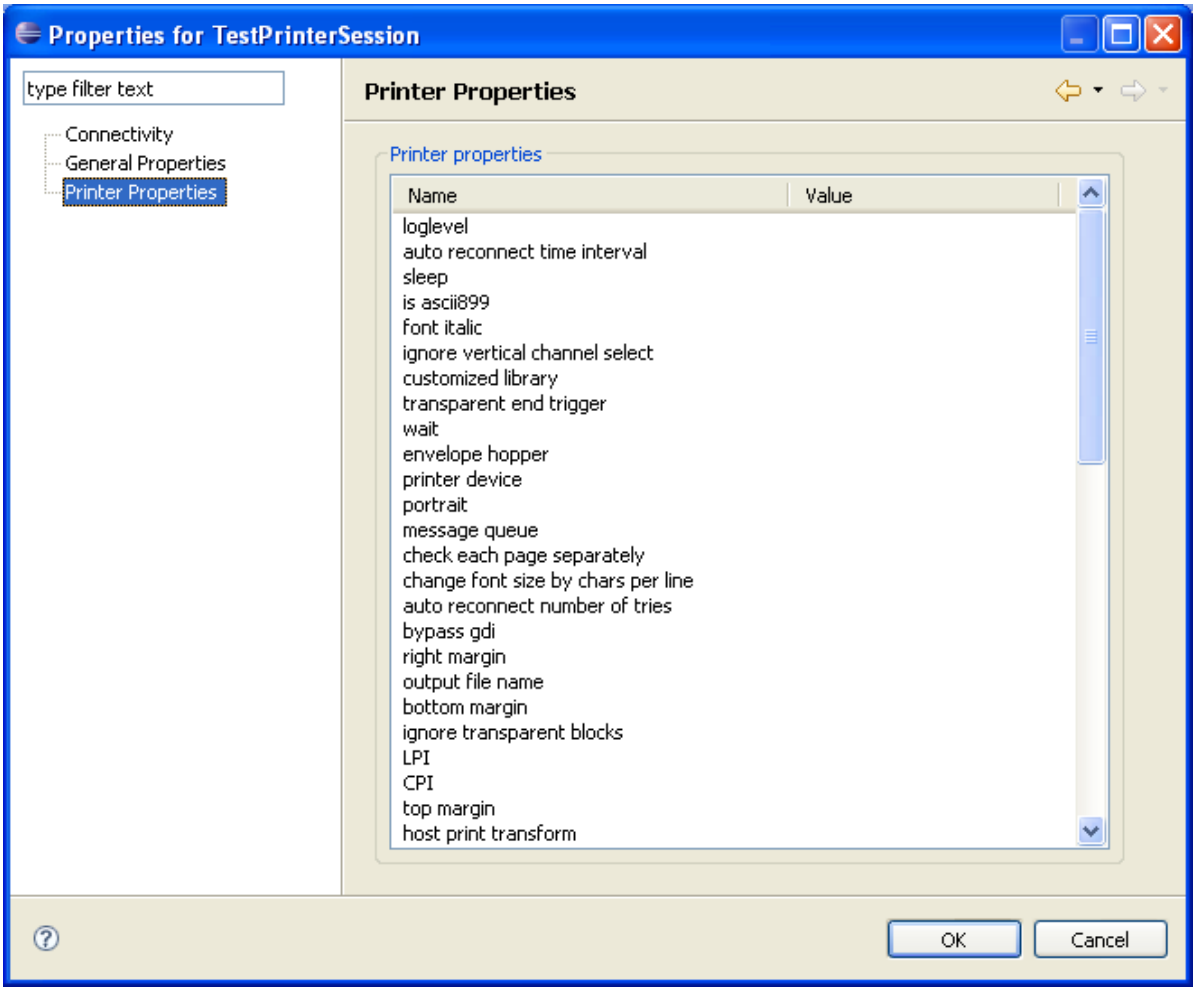
Configuring and Testing Various Printing Jobs

There are a number of parameters which enable defining the printing job output to suit the specific customer requirements (Refer to the Printlet API).

To configure and test printing jobs:

1 Connect the Printlet to the host (refer to Connecting the Printlet to the Host).

2 In the Printer Session Properties, set the Printlet parameters according to the parameters of the emulation you are currently using (for example, the number of lines to be printed per page, CPI, LPI etc.).



3 Print a variety of typical jobs, such as 132 characters, landscape, portrait, a number of pages, barcode printing etc.

4 Change the printer parameters to fine-tune the printing jobs and provide satisfactory results.

Tracing Printer Sessions

To trace a printer session:

1 To define that every printer session will be traced, open the application properties in ApplinX Designer, **Application Propterties>Printer tab**.

To define the trace file for a specific printer session definition, refer to **Getting Started with ApplinX Printer**.

2 Select the **Trace>File name** checkbox, and enter a file name

3 Select the **Session ID** checkbox. This option will create a separate trace file for each user.

4 Select the **Time** checkbox, to create a separate trace file for each session created by a different user.

III ApplinX Entities

11 Working with Entities

Creating a New Entity

New entities can be created via the **File>New>Entity** menu item. When creating a new entity, you are required to enter a name.

Note: The entity's name can only include a-z, A-Z, 0-9, or "." and must not start with a digit (0-9) or ".".

Editing an Entity

Edit the entity by double clicking on the entity in ApplinX Explorer. A single click on an entity will display the entity's properties in the properties area. The description of the entity can be edited in the Properties area.

Deleting, Copying and Pasting an Entity

Delete, copy and paste an entity by right-clicking on the entity in ApplinX Explorer and selecting the relevant option.

Locating an Entity

Locate/find and open an entity by right-clicking on the repository in the ApplinX Explorer and selecting **Open Entity...**, or by selecting **Open Entity...** in the **Navigate** Menu. Use ? to substitute a single character, and * to substitute a string.

Creating a New Folder

As an ApplinX user, you may like to divide applications into folders according to the specified interests and needs, such as development teams, sub-applications etc. The same folder contains different types of entities and sub-folders varying according to each application. This allows you greater flexibility organizing your entities. A new folder can be created via the **File>New>Entity>New Folder** menu item.

Copying Information from Host Screens

The Designer has built-in access to the host session, this session can be used to make the definition of entities easier. This is achieved using the Session's Copy from session feature. To copy from the session, mark the relevant string of data in the host's screen, and next to the relevant field click the Copy from session icon  . The string and/or position are copied.

Copying Entities

Entities can be copied between from folder to folder within an application and/or between applications/servers. This can be performed by dragging and dropping the relevant entity or by using the Copy/Paste actions. When copying between applications, the pasted entities will be placed in the root node of the repository.

Note: When coping an entity to an application that has the same entity name, the existing entity will be overwritten by the new one. The timestamps of the new entity (for example creation and modification date) also overwrite the values of the original entity. These changes take effect immediately, not after reloading the application.

Opening/Finding a Specific Entity

You can search and open a specific entity in the Editor area by right-clicking on the Repository node and selecting **Open Entity (Find)...**. The search can be filtered by entity and you can determine to search within a specific folder. You can sort the results by created/modified timestamp.

Renaming an Entity

You can rename an application or entity (though not a field) by selecting it in the ApplinX Explorer and then either selecting **Rename...** from the **File** menu or by right-clicking on the application/entity and selecting **Rename...**.

Renaming Fields

Field renaming is performed via the Field Mappings tab in the Screen Editor where the relevant field is mapped. Before renaming a field, you may want to see which other entities refer to this field. This is done by clicking on the **Find Referring Entities** icon next to the **Field name** field. To rename a field, click on the **Edit Entity** icon next to the **Field name** field, and access the field editor. In the field editor click, on the **Rename Field** button and rename the field.

Note: A field which does not have references can be edited directly in the **Field name** field in the screen editor.

Viewing an Entity's References

ApplinX entities may be interrelated with each other. The ApplinX Server maintains tables that keep track of entities that refer to other entities. For example, a field entity: the screens it is mapped to, the procedures it is an input or output parameter of, or the paths that employ it as an input / setting variable / test by field. This feature allows you to keep in check the database and avoid situations in which deleting one entity makes another entity non valid.

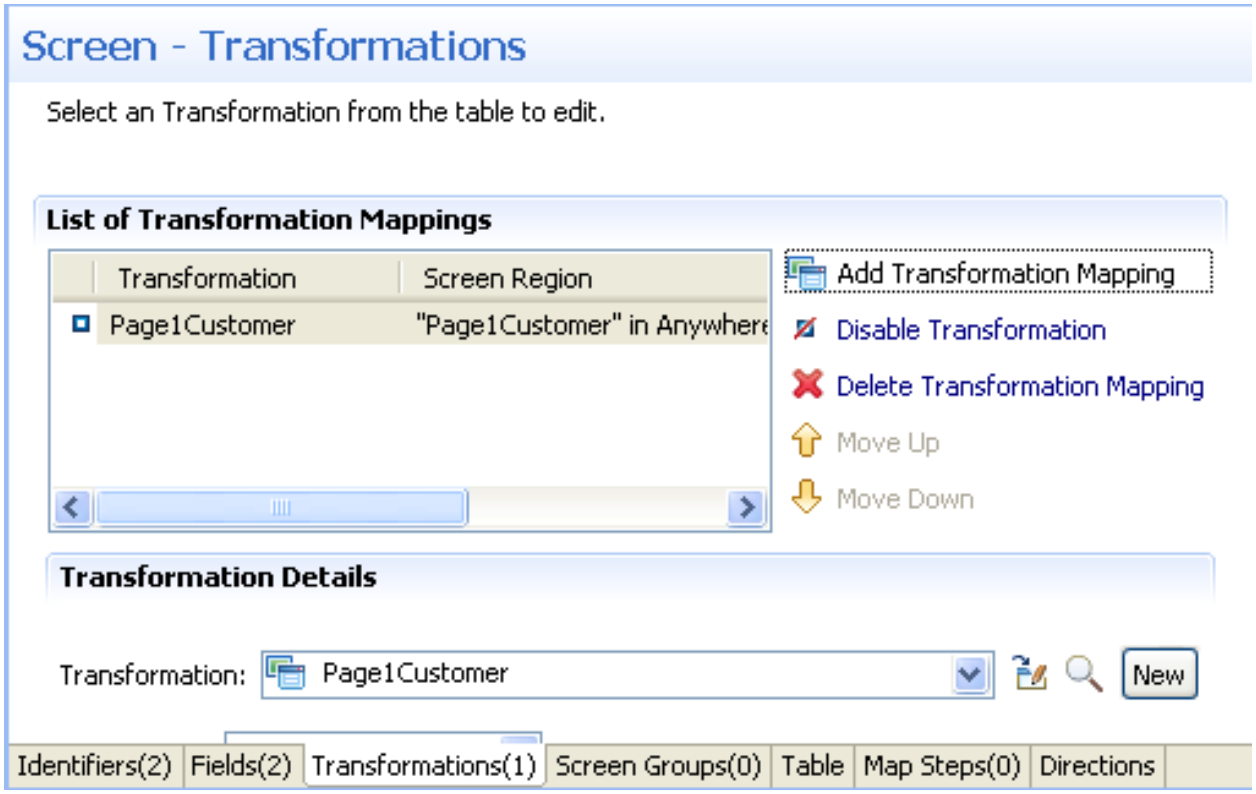
To view an entity's references

- In the ApplinX Explorer, right-click on the entity and select **Find Referring Entities**.

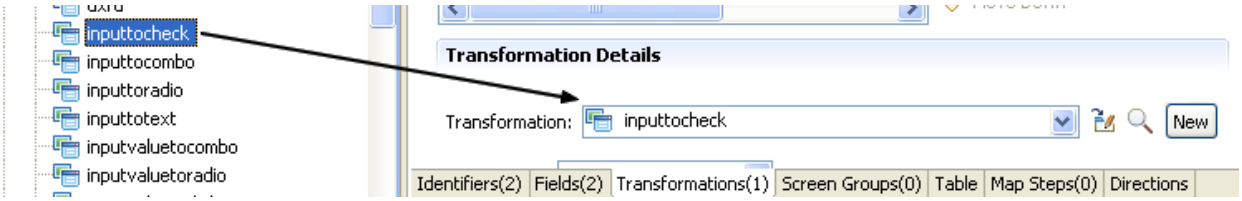
  The references are displayed in the reference view. Double-click on an entity to edit it.

Selecting Entities in Editor Page Fields

When working in an entity Editor such as the Screen editor, the Transformation editor or any other editor, you are sometimes required to select an entity from the application's repository. This is possible either by clicking on the drop-down arrow which appears on the right of the relevant field, or by dragging and dropping the entity from the Applinx Explorer tree into the field. For example, in the Transformation tab of the Screen editor, after adding a new transformation mapping, you are required in the Transformation Details area to select the transformation.



You can either click on the arrow next to the transformation field, or drag a transformation from the ApplinX Explorer tree



12 Screens

What is an ApplinX Screen?

A single ApplinX screen represents a corresponding screen in the host. An ApplinX screen is usually defined using unique text identifiers from the host screen but it is also possible to use other identifiers such as the position of the cursor when the screen is loaded or field attributes (Protected, Intensified or Hidden). When navigating in the host application, ApplinX tries to find the screen entity that matches the host screen it encounters. Once a match is found, ApplinX can use the data and definitions of the host screen in its various components. Sometimes the screen image needs to be updated, click on the Capture image icon to update the image. The following guidelines will help you determine which screens to identify:

- Screens that will be customized in a Web application (will be generated as web pages) as ApplinX uses identified screens to correctly load the customized Web page.
- Screens that will be part of navigation paths (recommended for easier path definition).
- Screens that will contain application field mappings.
- Any important screen in the host application.

In this section we will cover:

Creating Screens

To allow the ApplinX Server to recognize the corresponding host screen; each screen must be identified by giving it a unique name, and defining the screen's identifiers. In addition, when relevant, you may want to map fields to the screen, apply transformations to the screen, define screen based tables, associate screen groups to the screen and define steps in the application map.

ApplinX screens can be created manually, via the Session View (**Create a New Screen using the Session View**), by importing application host maps such as Natural, BMS and MFS (Importing an Application's Maps), or automatically or semi-automatically, using identification definitions (Changing the Screen Definition Mode).

- **Creating Screens via the Session View**
  - **Manually**
    - **Define Identifiers**
    - **Map Fields**
  - **Using Screen Creation Definitions**: Screens can be identified and created quickly using general screen creation definitions.
    - **Using Screen Creation Definitions: Manually**: The *New Screen* wizard is displayed when clicking on the **Identify New Screen** icon. Manual identification can be used with or without Screen Creation Definitions.

- **Using Screen Creation Definitions: Semi-automatically**: The *New Screen* wizard will automatically be displayed for each unknown screen that matches a Screen Creation Definition. This screen will include basic screen definitions, which are defined in the Screen Creation Definition.
- **Using Screen Creation Definitions: Automatically**: A screen will automatically be created for each unknown screen that matches a Screen Creation Definition. This screen will include basic screen definitions, which are defined in the Screen Group.

- **Automatically identify screens using the Trace Files (GCT) wizard**
- **Importing a Host Application's Maps**

Creating Screens via the Session View

Manually

Screens can be created via the **New Entity** menu or from within the session view.

Note: The screen must be displayed in the session view when creating the new screen.

To create a screen via a session:

1. In the Emulation session, navigate to the relevant screen.
2. Select text in the emulation screen.
3. Click on the Identify Screen button  . The New Screen wizard is displayed, with a suggested name displayed in the Name field. Enter a suitable description and determine the folder where the screen is to be located. Click **Finish**. You have now created a new screen entity in the repository.

Define Identifiers

For the ApplinX Server to recognize a host screen, you need to identify the screen in ApplinX. The screen is identified using "identifiers". Different identifier types include identifying specific text on the screen, the screen cursor position or the field attributes. The ApplinX Server analyzes the current screen and tries to match it to all of the screen identification strings stored in the application repository (database). For a screen to be recognized, all its identifiers must be matched. You may define an unlimited number of identifiers.

Text Identifiers: Use text identifiers to identify a screen by selecting specific text that appears on the screen. Use the Capture from Screen feature to identify a string of text accurately from the host screen and insert this string into the text box.

To define a text identifier:

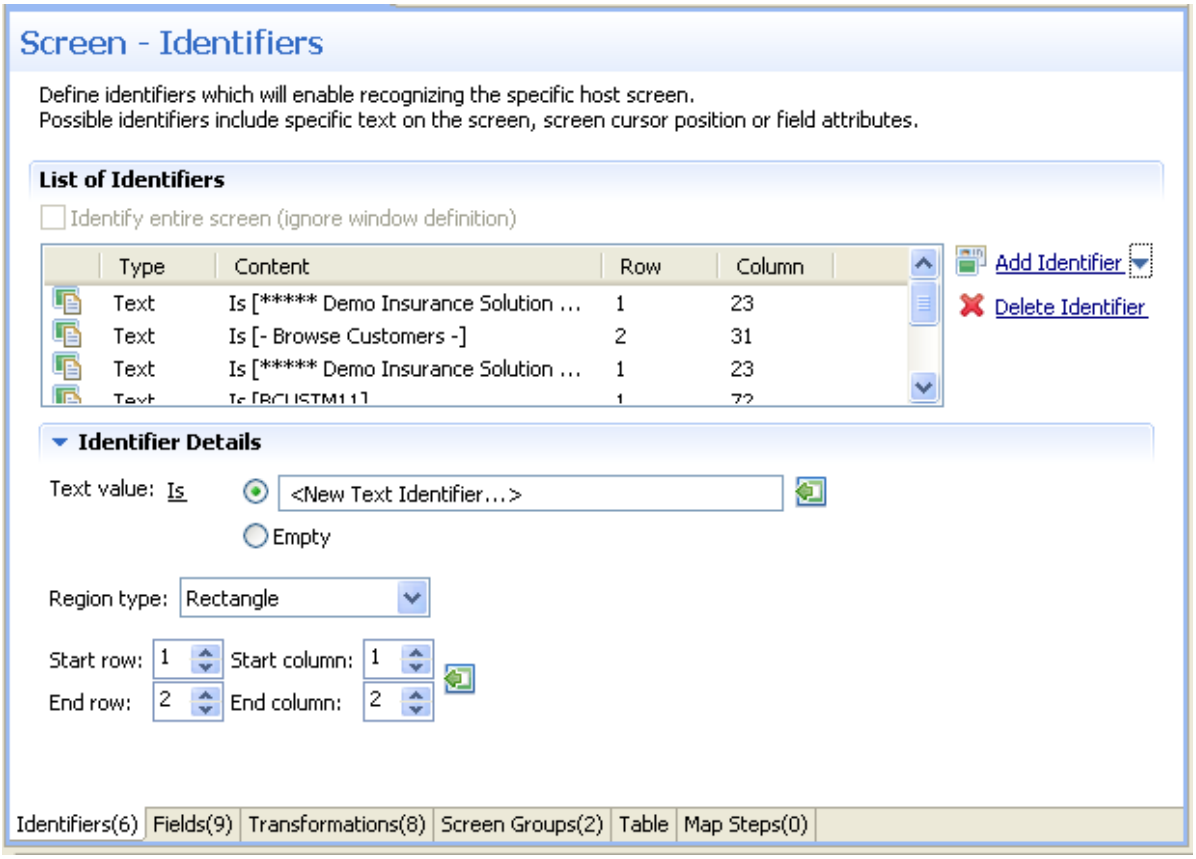
1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 Determine whether to ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used. This check box is only enabled when the screen has a window.

> **Note:** When changing the state of the **Ignore entire screen (ignore window definition)** check box, you must manually edit the identifiers and mappings to support the new positions.

4 Click the arrow on **Add Identifier** and select **Text**. Another row will be added to the list of identifiers.



5 Select **Is** (default) in order to match the exact text on the host screen. Alternatively, select **Is NOT** if any text other than the specified is suitable as a screen identifier. Check **Empty** to identify an empty space on the host screen. This clears the **Text** text box.

6 Select **Rectangle** to indicate that the identifier should be searched for within the defined rectangle, **Position**, to indicate that the identifier must start at a specific position or **Anywhere on screen** to indicate that the identifier may be anywhere on the screen.

7 When selecting Rectangle, define the rectangle range (start and end row and column.). When selecting Position, select the row and column.

8 It is also possible to add an identifier by selecting the area in the host screen, and then clicking Add Identifier. When selecting an area which is a rectangle, identifiers will be added for each and every row of the selected rectangle.

Cursor Position Identifiers: Use the Cursor Position identifier to identify a screen by the cursor position when the screen is loaded.

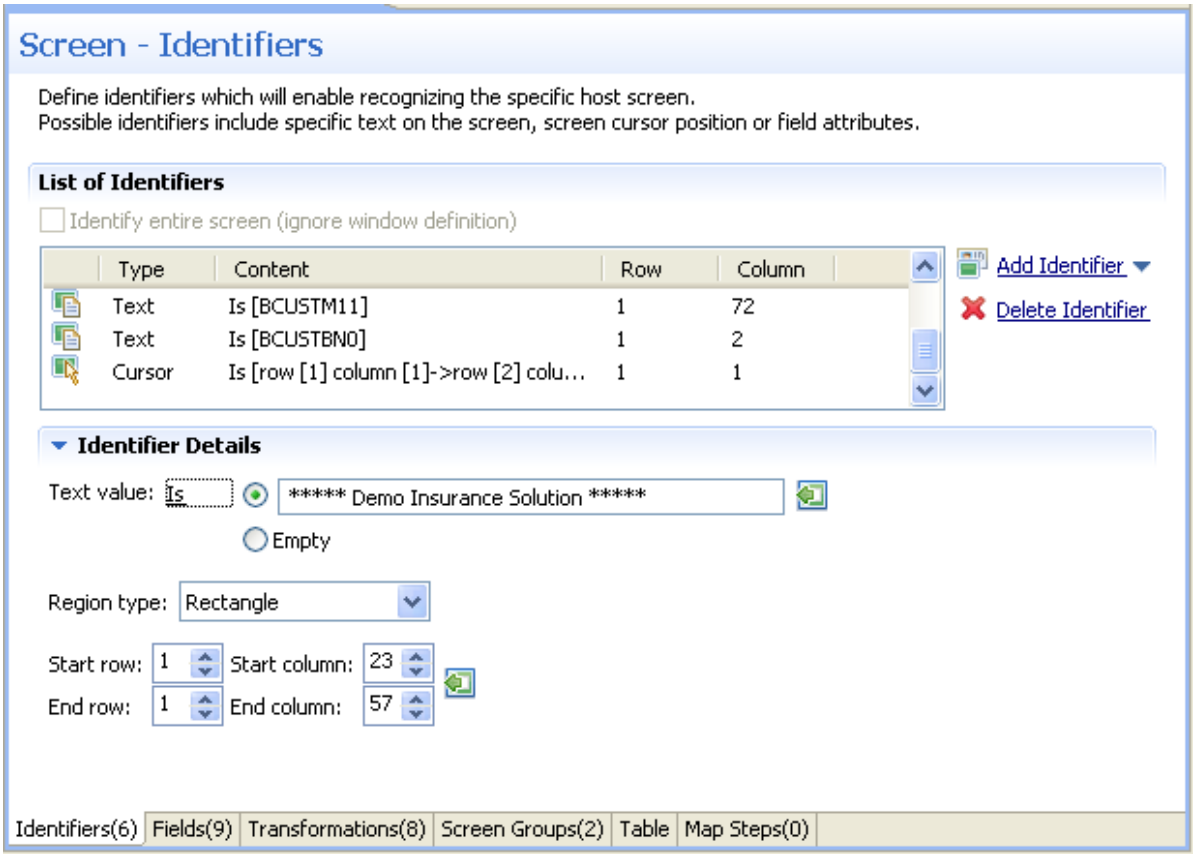
To define a cursor position identifier:

1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 To ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

4 Click the arrow on **Add Identifier** and select **Cursor Position**. Another row will be added to the list of identifiers.



5 Select **Is** to indicate that the cursor is positioned within the region area details that follow. Alternatively, select**Is NOT** to indicate that the cursor is not positioned within the region area details that follow.

6 Select **Rectangle** to indicate that the cursor position should be searched for within the defined rectangle or **Position**, to indicate that the cursor position must start at a specific position

7 It is also possible to add an identifier by selecting the area in the host screen, and then clicking Add Identifier.

Attribute Identifiers: Use the Attributes identifier to identify a screen by the host's attributes. Attributes of the host screen may be Protected, Intensified or Hidden.

To define an attribute identifier:

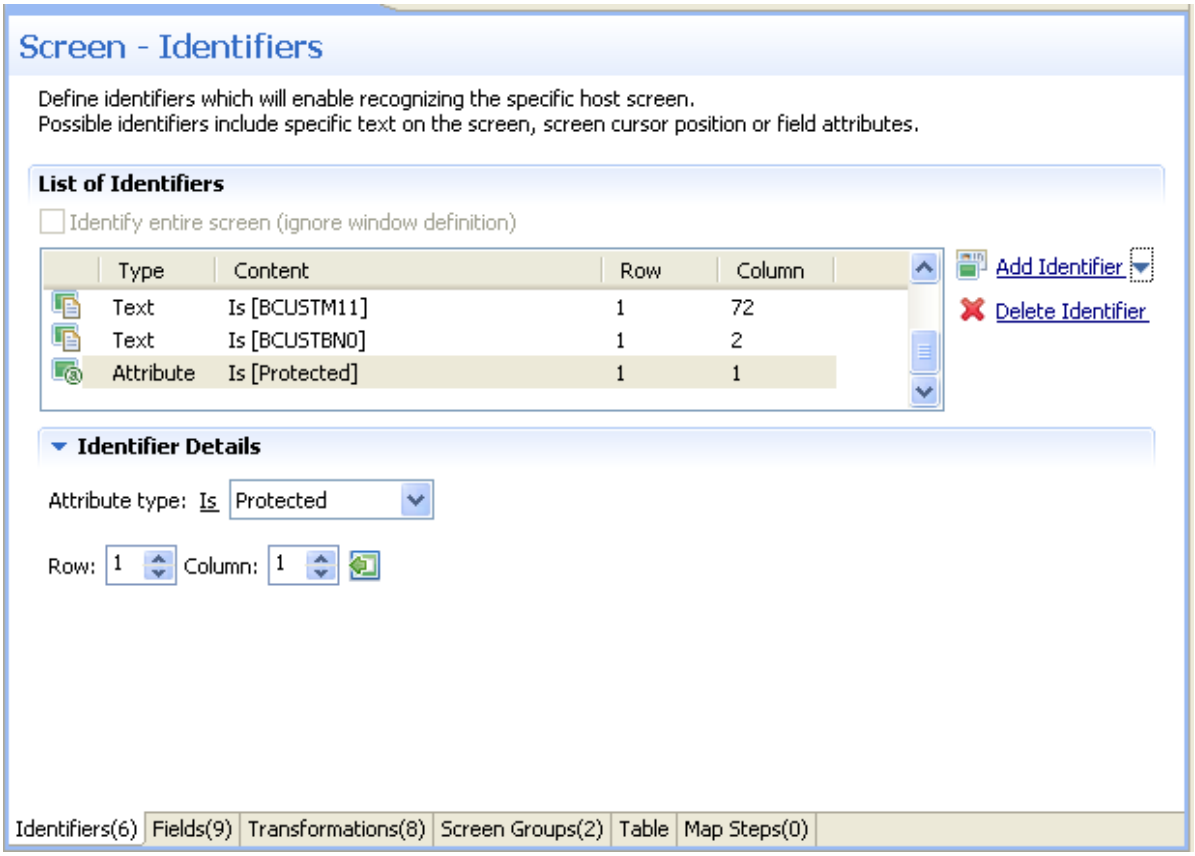
1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 To ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

Note: When changing the state of the **Ignore entire screen (ignore window definition)** check box, you must manually edit the identifiers and mappings to support the new positions.

4 Click the arrow on **Add Identifier** and select **Attribute**. Another row will be added to the list of identifiers.



5 Select whether the Attribute Type **is/is not Protected**, **Hidden**, **Intensified** or **Reversed video**.

6 Enter values or use the Capture from Screen feature to indicate a specific location of the attribute by its **Row** and **Column**.

7 It is also possible to add an identifier by selecting the area in the host screen, and then clicking Add Identifier.

Window Identifiers: Use the Window identifier to identify a screen by determining whether it is or is not a window. In Natural UNIX, it is possible to identify a screen by determining whether it is a window and whether it has specific text in the title.

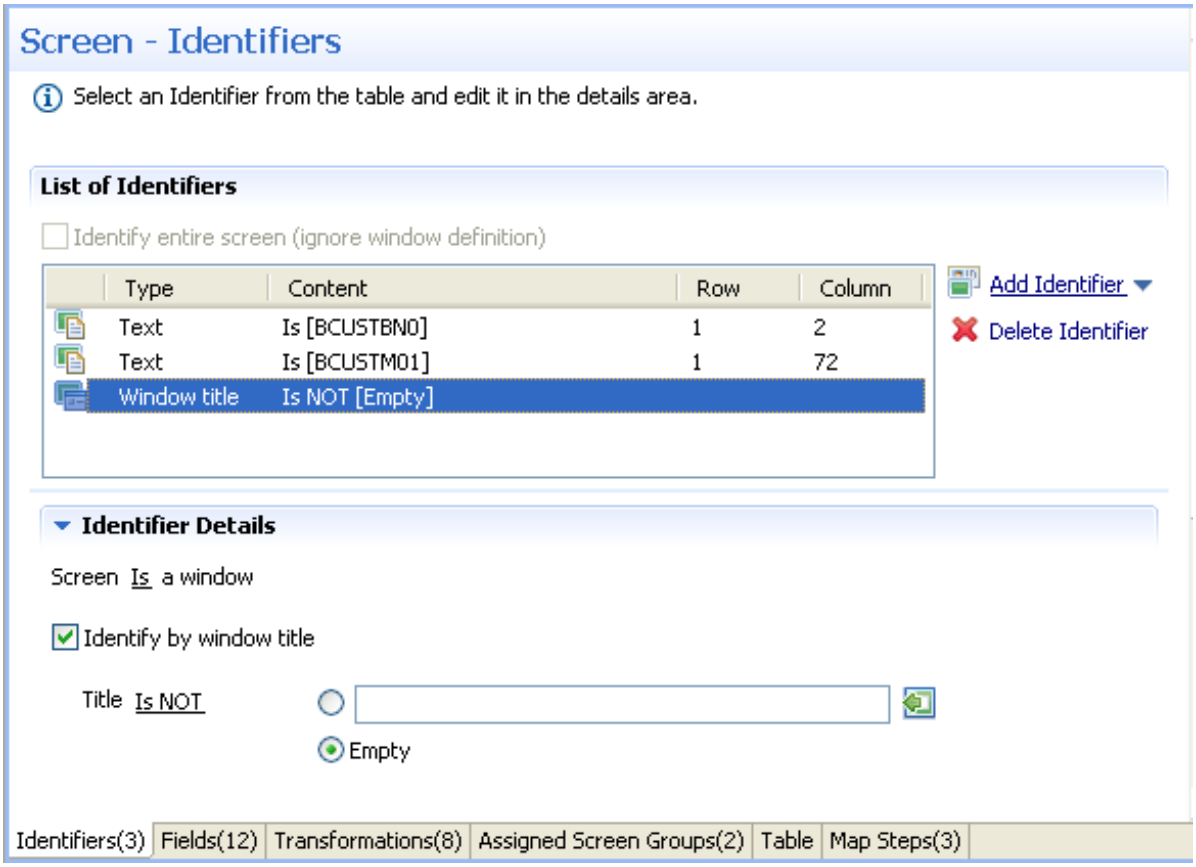
To define a window identifier:

1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 To ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

4 Click the arrow on **Add Identifier** and select **Window**. Another row will be added to the list of identifiers.



5 Select **Is** to indicate that the screen is a window. Alternatively, select **Is NOT** to indicate that the screen is not a window.

6 For Natural UNIX hosts: Select the check box to indicate to identify by the window title. You can enter text for the title or select the text in the session view and click the Capture from screen icon. You can also identify a screen which is a window and does not have a title by selecting the Empty radio button.

Map Fields

What is an ApplinX Field?

When a screen is identified, field entities can be mapped to it. A field makes relating to a string on the host screen much simpler. Fields offer an enhanced way of referring to host fields - by their name rather than by their position. In addition, mapped application fields simplify future application maintenance - when a field changes in the original host application, updates only need to be made to the application field definition. The following guidelines will help you determine which fields to identify:

- Fields that will be referenced in code (when binding Web page controls to host fields by name).
- Fields that will be part of navigation paths.
- Fields that will be used as Procedure input and output attributes to expose host transactions as Web services.
- Fields that will be used when creating generated Web tables.
- Recommended: all input and output fields (not constants).
- Any important field in the host application.

An application field makes relating to a string on the host screen much simpler. Fields offer an enhanced way of referencing host fields - by their name rather than by their position. Using fields has many advantages:

- A field's ID is meaningful regarding the contents of the field, unlike a numeric position.
- You define a field only once, and later can reference that definition in all of your code - knowing you always mean the same definition.
- If changes in the host application occur in the future, the only place that requires to be changed is the field definition.

Field properties can be accessed by right-clicking on the Repository node and selecting **Open Entity (Find)...** and then searching for the specific field.

In the Field Editor, it is possible to set the Input Mask (Defines the values that can be entered in the field) and the field type (numeric or alphanumeric).

Input Masks

An input mask consists of literal characters along with special characters that together determine the value that may be entered into input fields. When a user defines a mask on an input field, client side validation will be carried out on the input format. Common usage is in date/time and number or currency fields.

To implement this feature you must change the web application configuration in the configuration editor:

1 Open a new browser and run your Web application.

2 Click on the Configuration link. The Configuration Editor will be displayed.

3 In the **Instant** node select the **Reflect emulation behavior** check box.

The following table shows some useful input mask definitions and examples of values you can enter. Refer to Valid input Characters for details on the codes used to create input mask definitions.

Input Mask Definition	Example Values
(000) 000-0000	(206) 555-0248
(999) 999-9999	(206) 555-0248 ( ) 555-0248
(000) AAA-AAAA	(206) 555-TELE
#999	-20 2000
>L????L?000L0	GREENGR339M3 MAY R 452B7
00000-9999	98115- 98115-3007
L?????????????	Maria pierre
ISBN 0-&&&&&&&&&&-0	ISBN 1-55615-507-7 ISBN 0-13-964262-5

Valid Input Mask Characters

ApplinX interprets characters in the Input Mask property definition as shown in the following table. To define a literal character, enter any character other than those shown in the table, including spaces and symbols. To define one of the following characters as a literal character, precede that character with a "\".

Character	Description
0	Digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed).
9	Digit or space (entry not required; plus and minus signs not allowed).
#	Digit or space (entry not required; blank positions converted to spaces, plus and minus signs allowed).
L	Letter (alphabetic, entry required).
?	Letter (alphabetic, entry optional).
A	Letter or digit (entry required).
a	Letter or digit (entry required).
&	Any character or a space (entry required).
C	Any character or a space (entry optional).
\	Causes the character that follows to be displayed as a literal character. Frequently displayed any of the characters listed in this table as literal characters (for example, \A is displayed as just A).

When a screen is identified, field entities can be mapped to it. A field makes relating to a string on the host screen much simpler. Fields offer an enhanced way of referring to host fields - by their name rather than by their position. In addition, mapped application fields simplify future application maintenance - when a field changes in the original host application, updates only need to be made to the application field definition. The following guidelines will help you determine which fields to identify:

- Fields that will be referenced in code (when binding Web page controls to host fields by name).
- Fields that will be part of Path Procedures.
- Fields that will be used as Procedure input and output attributes to expose host transactions as Web services.
- Fields that will be used within tables.
- Recommended: all input and output fields (not constants).
- Any important field in the host application.

There are limitations when defining a mapping. A mapping cannot:

- Start in an unprotected field and end in a protected field, or vice versa.
- Start with an attribute byte.
- Overlap another mapping on the same screen.
- Start in one line and end on another line.
- Start on one line and end off the screen's limits.

Fields can be mapped according to their position in the screen or according to their leading label - dynamic field mapping (do not set the same field to be mapped by position and by its leading label).

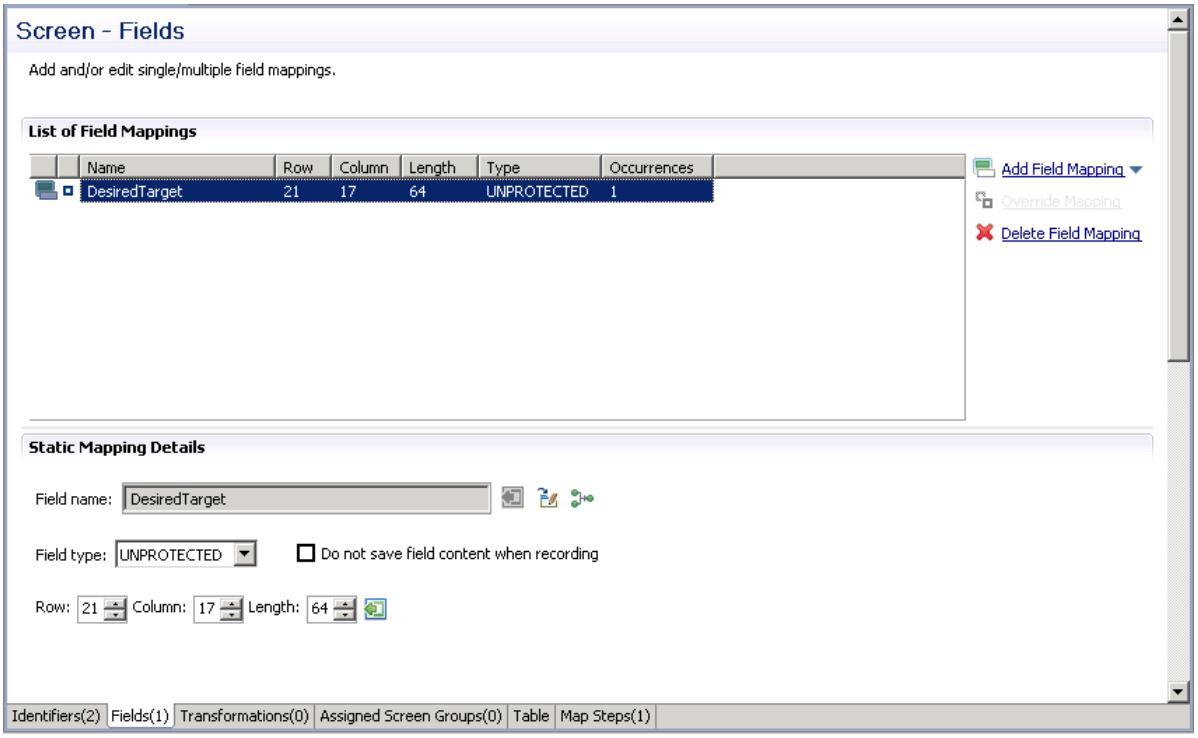
Mapping according to leading labels (dynamic mapping) is particularly useful in the following cases:

- The application has fields that are dynamically drawn on the screen.
- Host applications are sometimes changed and items can be moved. Using dynamic field mappings, the fields will continue to be mapped and identified and the application will not be affected.
- Using dynamic field mappings enables more flexibility when using Screen Groups, as Screens which include the same field, can be associated with the same Screen Group even when the field is located in a different position.

To map application fields according to their position in the screen:

1 Click the **Fields** tab to view and/or modify the application fields in this screen.

2 Click the arrow next to **Add Field Mapping**, and select **Single Mapping** or **Multiple Mapping** to add a new field mapping.



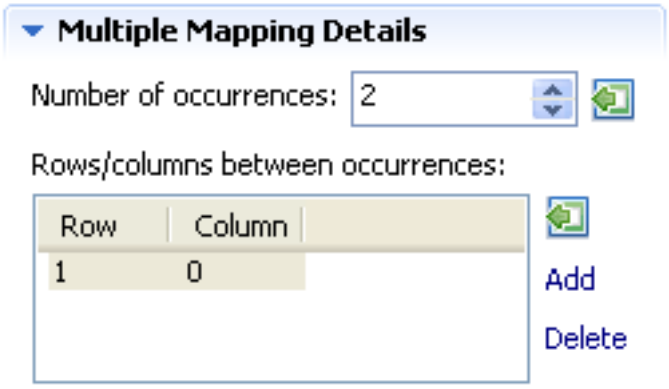
3 In the **Field name** field, either enter a field name or enter the first letters of a field name and then click CTRL and SPACE to display relevant fields.

4 Select the **Protection type**: **Protected** (cannot be edited in host), **Unprotected** (editable in host) or **Both** (may sometimes be editable and sometimes not).

5 The **Do not save field content when recording** check box enables not saving data which you do not want to be displayed when viewing the recorded trace file.

6 The **Row** and **Column** fields indicate the position of the field on the screen. The values of these fields can be changed using the **Capture from Screen** feature or by entering values in the fields.

7 When adding a multiple field mapping, determine the number of occurrences and the rows and/or columns between occurrences.



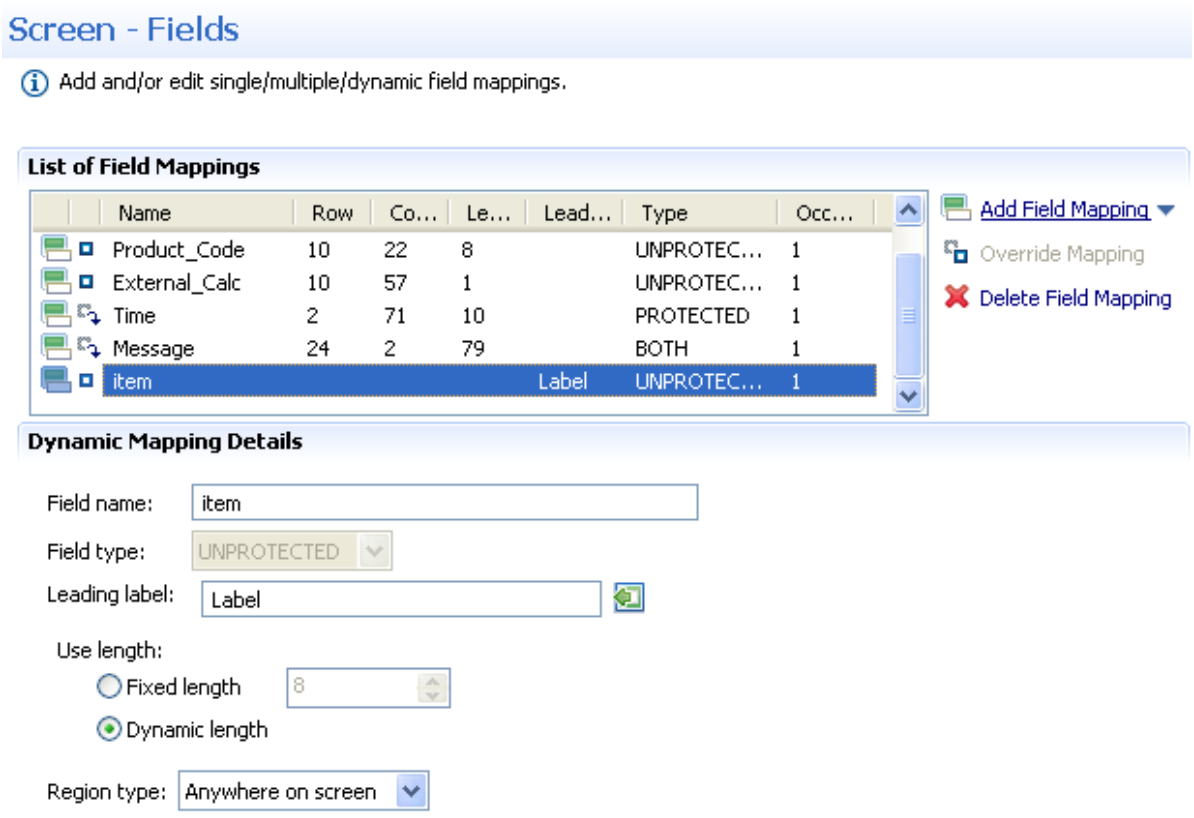
Note: You can automatically map all of the unprotected fields in the screen by clicking on the **Automatically identify unprotected fields** icon.

To map application fields according to their leading label (not available in right to left and character mode applications):

In some screens, the position of a field may vary. It is therefore necessary to map some fields in such a way, that even if they appear in a different position, they will still be recognized and mapped. This is possible by defining the label near the field (the label must be to the left of the field) and identifying the field according to this label. This mapping type is called "Single Dynamic" as the mapping position changes dynamically according to the leading label. Refer to Dynamic Field Mapping Limitations.

1 Click the **Fields** tab to view and/or modify the application fields in this screen.

2 Click the arrow next to **Add Field Mapping**, and select **Single Dynamic** to add a new field mapping.



3 In the **Field name** field, either enter a field name or enter the first letters of a field name and then click CTRL and SPACE to display relevant fields.

4 Select the field protection type: Unprotected, Protected or Both. Prefer to set the field type either as **Protected** or **Unprotected** when you are sure of the type of the dynamic field. Use **Both** when you are not sure of the type of the dynamic field.

5 Determine the leading label: the leading label is the label which appears before the field. This label is inserted as the Field name and can be edited. The label you define should be as accurate as possible (do not use strings which normally appear within the field).

6 Determine the method to use to search for the label:

- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note: When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

7 Define the region type in which the label will be searched for: anywhere in the screen, within a specified rectangle or within a specified row range. The **Row** and **Column** fields indicate the position of the label on the screen. The values of these fields can be changed using the **Capture from Screen** feature or by entering values in the fields.

8 Use length:

In Unprotected fields: Determine whether to recognize a field mapping according to the leading label and the length of the input field (Fixed length) or just according to the leading label (Dynamic length).

In Protected fields: The length defined here is the length of the mapping. The identified field is cropped to the specified length.

9 Define the region type in which the field will be searched for: anywhere in the screen, within a specified rectangle or within a specified row range. The **Row** and **Column** fields indicate the position of the field on the screen. The values of these fields can be changed using the **Capture from Screen** feature or by entering values in the fields.

Note: The region will be displayed in the Session View with a yellow frame, and the background of the screen within this region will be dotted.

Recommended guidelines for mapping fields according to their leading labels:

- It is preferable to define your regions as rectangles, that do not overlap each other, and whose size are minimal.
- The feature is best suited to locate dynamic fields which change their location vertically so that labels are aligned vertically in one region, and fields aligned vertically on a region to the right of the labels.

Mappings inherited from screen groups

- Mappings inherited from screen groups are indicated with an icon. You can override this mapping to suit the current screen by clicking on the **Override** hyperlink. Hereafter, this mapping will be marked with the icon. You can restore the inherited mapping by clicking on the **Restore Inherited Mapping** hyperlink. Note that when selecting an inherited mapping, it is not possible to change the mapping properties. When selecting a local mapping, all properties can be changed. When selecting an overridden mapping, all properties can be changed, except for the name.

To delete a field mapping, click on the **Delete Field Mapping** hyperlink.

Note: It is not possible to remove mappings which are used in other entities, such as in procedures, transformations or tables.

Sorting

You can sort the contents of a **List of Field Mappings** table by field name or by any other column. Click the header of the column by which you want to sort. The contents are sorted in ascending order. Click again to sort in descending order. This feature is useful when detecting duplicate field entries. The sort will be cancelled when you close the screen entity.

VT Parameters

VT parameters are defined per application. Sometimes it is necessary to override these parameters for a specific field, for example when it is necessary to define that a field is a password type field. this means that characters are sent from applinX to the host, but only asterisks are retrieved from the host. in order to validate that the correct data has been received from the host, it needs to be indicated that this field's characters are hidden. This option is selected in the by opening the relevant screen in the Editor view and in the Fields tab, changing the VT Parameter's configuration (in the Host field behavior field). Additional parameters can also be configured here. Refer to Fields tab: VT Parameters section.

Using Screen Creation Definitions

Screen Creation Definitions are a set of basic definitions which are used to create a new ApplinX screen. These definitions include the default screen name, initial identifiers and determine whether input fields will be created. Creating a new screen using screen creation definitions reduces the complexity and the time required to identify new screens.

How does it work?

To use this feature, at least one set of Screen Creation Definitions must be created within a **Screen Group** (in the dedicated tab). When navigating within a session, and reaching an unidentified screen, ApplinX searches for Screen Groups which match the unidentified screen, and then creates a new screen based on the screen group's identifiers and on parameters defined in the creation definition configured in the Screen Group. This process can be initiated in three different modes:

- Automatically: A screen will automatically be created for each unknown screen that matches a screen group. This screen will include basic screen definitions, which are defined in the screen group.
- Semi-automatically: The New Screen Wizard will automatically be displayed for each unknown screen that suits a screen group. This screen will include basic screen definitions, which are defined in the screen group.
- Manually: The New Screen Wizard is displayed when clicking on the Identify New Screen icon.

The mode is set in the session view by clicking on the Change screen creation mode icon in the tool bar. Refer to Changing the Screen Definition Mode.

Using Screen Creation Definitions: Manually

The difference between creating a screen in the manual mode and creating a screen in the semi-automatic mode, is that in the manual mode, when you reach an unidentified screen, you are required to click on the Identify New Screen icon in the Session view. You can then select from a list of Screen Groups relevant to this screen, the screen group which has screen creation definitions which you would like to use (you can select to create a new screen without using screen creation definitions by clicking on **No** in the *Use a Screen Creation Definition* dialog box). By default the list of screen groups which are relevant to this screen are displayed. It is also possible to show and select a screen group from all the application screen groups that have screen creation definitions. If you want ApplinX to always use the **Screen Group with the highest priority**, select the **Don't ask again** check box. It is possible to display this dialog box again by changing the **Always use the highest prioritized screen group** check box in **Windows>Preferences>Software AG>ApplinX**.

The new screen has a number of basic definitions such as the screen name and a number of identifiers. These are determined according to the definitions in the **Screen Creation Definition** tab in the relevant Screen Group entity (You can select to create a new screen without using screen creation definitions by clicking on **No** in the *Use a Screen Creation Definition* dialog box). The screen name is determined according to the screen name position definition in the Screen Creation Definition tab. The list of identifiers is determined according to the Screen Group identifiers, as well as the strings which appear in the list of identifier locations in the Screen Creation Definition tab. To automatically create input fields in the new screen, select **Automatically create input fields**. The labels near the input fields are used as the input field names.

To manually create screens using screen creation definitions

1. Within the relevant Screen Group, **create a Screen Creation Definition**.
2. Ensure that in the session view the selected mode is Manual Screen Creation: in the Session view click on the Screen Creation Definition mode icon and select Manual Screen Creation.
3. Navigate within a session.
4. Once a screen group is recognized within an UNKNOWN screen, click on the Identify New Screen icon. The *Use a Screen Creation Definition* dialog box is displayed. Here you can determine whether to base the new screen on a Screen Creation Definition or not. Click **No** to create a new screen without using Screen Creation Definitions.
5. This dialog box displays, by default, the screen groups which have screen creation definitions and are relevant to the UNKNOWN screen. You can display all the screen groups which have screen creation definitions and select a different screen group. Click **Yes.**
6. The New Screen wizard is displayed. The default name is taken from the text which appears in the screen name position definition in the Screen Creation Definition tab. Click **Finish.**
7. The new screen is displayed in the Editor and is listed in the repository.

Refer to **Automatically Creating Screens** .

Refer to **Semi-Automatic Creation of Screens**.

Using Screen Creation Definitions: Semi-automatically

Similar to the automatic mode, when navigating in a session and a screen group is recognized, and the current screen has not yet been identified, the process of creating a new screen entity is initiated. The new screen has a number of basic definitions such as the screen name and a number of identifiers. These are determined according to the definitions in the **Screen Creation Definition** tab in the relevant Screen Group entity (You can select to create a new screen without using screen creation definitions by clicking on **No** in the *Use a Screen Creation Definition* dialog box). The screen name is determined according to the screen name position definition in the Screen Creation Definition tab. The list of identifiers is determined according to the Screen Group identifiers, as well as the strings which appear in the list of identifier locations in the Screen Creation Definition tab. To automatically create input fields in the new screen, select **Automatically create input fields**. The labels near the input fields are used as the input field names. As there may be more than one Screen Group which has Screen Creation Definitions, it is necessary to determine which screen group is to be used. This is done in the *Use a Screen Creation Definition* dialog box which is displayed once the unknown screen is reached. By default the list of screen groups which are relevant to this screen are displayed. It is also possible to show and select a screen group from all the application screen groups that have screen creation definitions. If you want ApplinX to always use the **Screen Group with the highest priority**, select the **Don't ask again** check box. It is possible to display this dialog box again by changing the **Always use the highest prioritized screen group** check box in **Windows>Preferences>Software AG>ApplinX**.

To semi-automatically create screens using screen creation definitions

1. Within the relevant Screen Group, **create a Screen Creation Definition**.
2. Ensure that in the session view the selected mode is **Semi-Automatic Screen Creation**: in the Session view click on the **Screen Creation Definition** mode icon and select **Semi-Automatic Screen Creation**.
3. Navigate within a session.
4. Once a screen group is recognized within an unidentified screen, the *Use a Screen Creation Definition* dialog box is displayed. Here you can determine whether to base the new screen on a Screen Creation Definition or not. Click **No** to create a new screen without using Screen Creation Definitions.
5. This dialog box displays, by default, the screen groups which have screen creation definitions and are relevant to the UNKNOWN screen. You can display all the screen groups which have screen creation definitions and select a different screen group. Click **Yes**.
6. The New Screen wizard is displayed. The default name is taken from the text which appears in the screen name position definition in the Screen Creation Definition tab. Click **Finish.**
7. The new screen is displayed in the Editor and is listed in the repository.

Refer to **Automatically Creating Screens** .

Refer to **Manually Creating Screens**.

Using Screen Creation Definitions: Automatically

When navigating to an unidentified (UNKNOWN) screen in a session and a screen group is recognized, a new screen entity is created. The new screen has a number of basic definitions such as the screen name and a number of identifiers. These are determined according to the definitions in the **Screen Creation Definition** tab in the relevant Screen Group entity. The screen name is determined according to the screen name position definition in the Screen Creation Definition tab. The list of identifiers is determined according to the Screen Group identifiers, as well as the strings which appear in the list of identifier locations in the Screen Creation Definition tab. When more than one screen group is relevant to the screen, the screen creation definition is based on the definition in the **screen group with the highest priority**. To automatically create input fields in the new screen, select **Automatically create input fields**. The labels near the input fields are used as the input field names.

To automatically create screens using screen creation definitions

1 Within the relevant Screen Group, **create a Screen Creation Definition**.

2 Ensure that in the session view the selected mode is Automatic Screen Creation: in the Session view click on the **Screen Creation Definition mode** icon and select **Automatic Screen Creation**.

3 Navigate within a session. New screens are automatically added to the repository.

Refer to **Semi-Automatic Creation of Screens**.

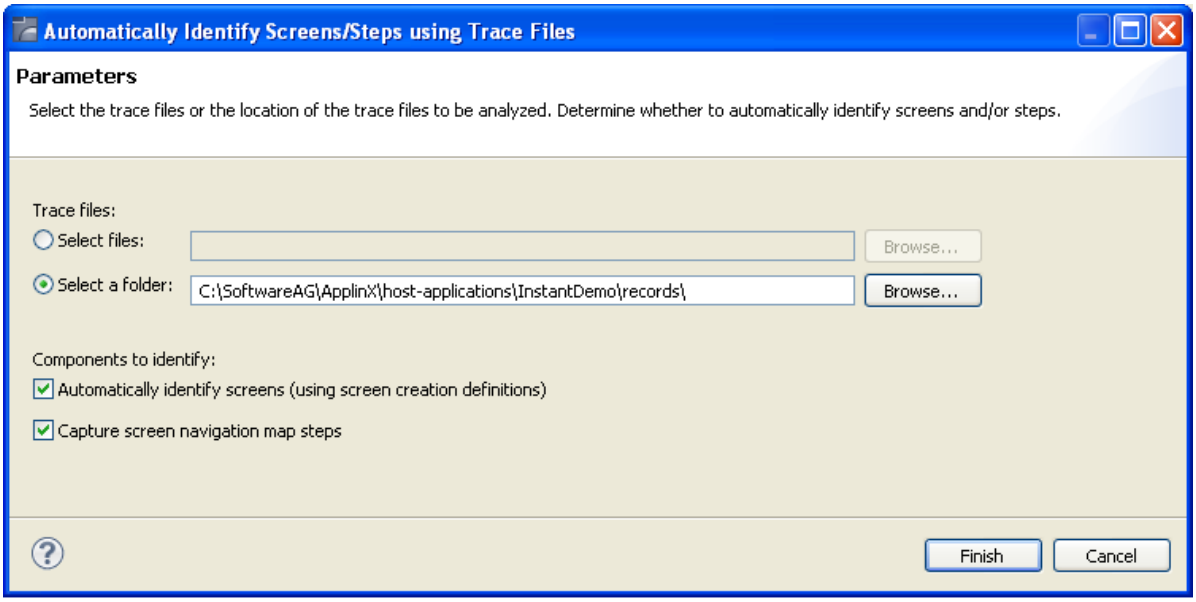
Refer to **Manually Creating Screens**.

Automatically Identify Screens using Trace Files (GCT)

Using trace (GCT) files, you can automatically identify screens. In order to do this, you must ensure that a Screen Group includes a **Screen Creation Definition**.

To automatically identify screens using trace files:

1 Right-click on the relevant application, and select **Automatically Identify Screens/Steps using Trace Files...**. The wizard is displayed.



2 Select whether to analyze files or a folder. Locate the files/folder.

3 Select the **Automatically identify screens** check box.

You can automatically identify steps by selecting the **Capture screen navigation map** steps check box. Refer to **Automatically Identify Steps using Trace Files (GCT)** for further details.

4 Click **Finish**.

The Console displays the outcome of the process.

Note: When identifying a new screen, the screen name is taken from the area you defined in the Screen Creation Definitions. If a screen with this name already exits, the newly identified screen's name will be the name as of the existing screen followed by an incremental number. If the area you defined in the Screen Creation Definitions is empty, the screen will be named: "Screen" and then an incremental number.

Importing a Host Application's Maps

Standard maps, such as Natural, BMS and MFS, are used in host applications to describe the structure of the host screen including the static text and input and output fields. ApplinX enables importing these application maps, saving time and effort spent on manually identifying screens and simplifying the maintenance process when changes are made in the host application. When importing application maps, a screen is automatically created from each map, minimizing errors that may occur when creating the screens manually, one by one. The ApplinX screen created includes identifiers (based on the static text in the map) and fields (based on the input and output fields in the map). ApplinX supports a number of different types of maps:

- NATURAL: Natural map support (from SYSTRANS/SYSOBJH or NaturalONE files).

- BMS: CICS basic map support.
- MFS: IMS message format service.
- SDFX: ApplinX generic map format, used for other standard maps. To create SDFX files refer to SDFX File Format Definition.
- SDF: Compatible with Software AG's JIS product.

The import map feature can be used to import an application's maps for a new application or to maintain and update previously imported maps. When updating previously imported maps, screen identifiers will be deleted and replaced, existing fields will be updated with their new positions and their references to other entities will be preserved. Fields that were previously imported, but no longer exist on the host will be deleted.

Note: Invalid entity names, such as names which include invalid characters such as "#" or begin with a digit, will be automatically corrected by omitting the invalid characters.

Note: The colors displayed in the screen image may sometimes be different than the colors that are displayed in the original host screen.

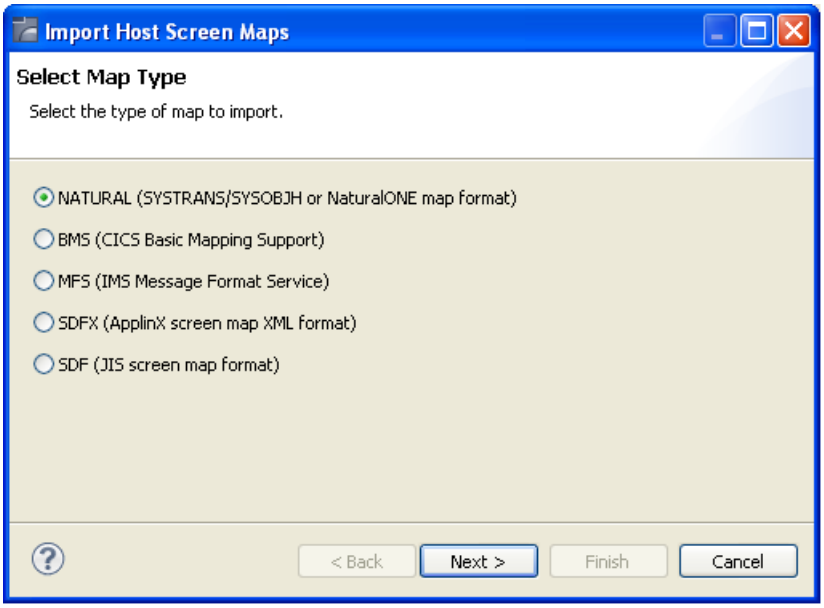
Maps can be imported either using the Import Host Screen Maps wizard or using a batch file (screen_import.bat in Windows and screen_import.sh in UNIX).

To import maps using the Import Host Screen Maps wizard

Ensure that the relevant ApplinX application and that the relevant map files are available.

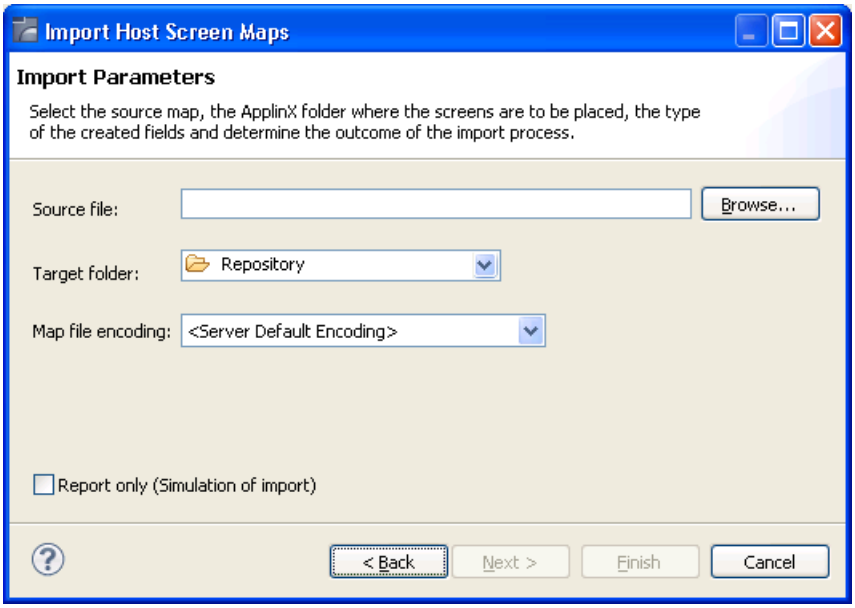
1 From the **Application** menu, select **Import Host Screen Maps...** or right-click on the repository node in the relevant application and then select **Import Host Screen Maps...**.

2 The *Import Host Screen Maps* wizard is displayed.



3 Select the type of map that is to be imported. Click **Next.**

4 The *Import Parameters* screen is displayed.



Select the source map file/s that you wish to import.

5 Select the ApplinX repository folder where the newly created screens are to be placed.

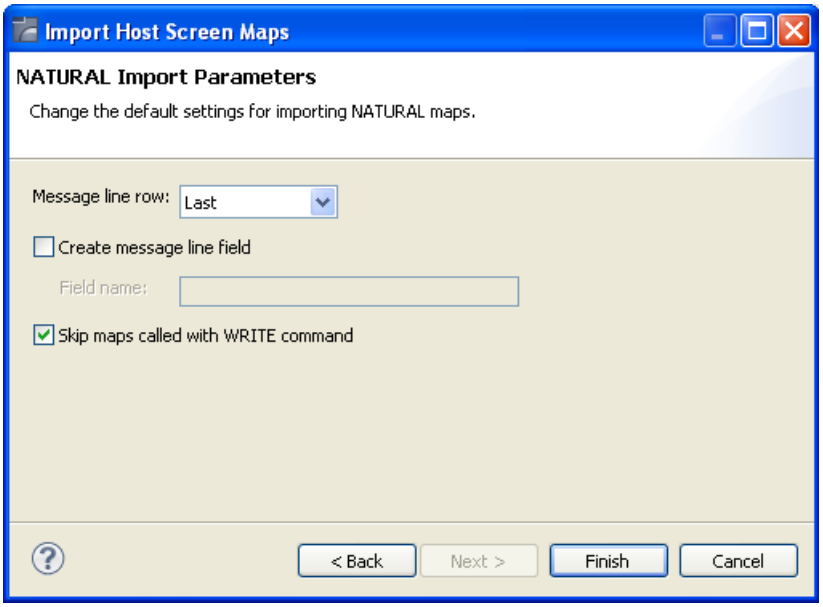
6 Select the map file encoding. By default, the server operating system default encoding is used.

Note: When importing maps from NaturalONE it is recommended to use UTF-8 encoding.

7 Determine whether you wish to simulate the import process, and create a report which details the import process or whether to actually import the maps and create the screens as well as the report.

8 If you selected to import a Natural map, you can click **Next** to configure the NATURAL Import Parameters, otherwise click **Finish** to complete the wizard.

9 NATURAL Import Parameters screen:



Message line row
Indicates where the error line is located - in the first line, one, two, three or four lines from the bottom or in the last line.

Create message line field
Selecting this will map a field in every screen, where a message can be displayed. When selecting this option, you are required to enter a name for the field.

Skip maps called with WRITE command
Select this in order not to generate maps that are called with the NATURAL WRITE command.

Click **Finish**.

The screens created appear in the directory you determined in the **Target folder** field. The names of the screens are identical to the map names.

The report is displayed in the Eclipse console and includes a list of the screens added as well as the fields and identifiers created/updated/deleted.

To import screens via a batch file (using the command prompt window):

1 Open a command prompt window.

2 Change the current directory to the ApplinX home directory.

3 Type screen_import.bat/sh followed by your required parameters. (The minus sign and letter should precede the value to distinguish between the parameters. The order of the parameters is not significant.)

Parameter	Description	Default
-a	ApplinX application name (Required parameter)	
-af	ApplinX target folder within the application repository.	Root folder
-mf	Message line field name (Natural maps only)	MessageLine
-k	Don't skip map with write command. (Natural maps only)	true (skip)
-e	Map file encoding: Includes all encodings which are supported by Java. **Note:** When importing maps from NaturalONE it is recommended to use UTF-8 encoding.	Server operating system default encoding
-f	File name, or directory name (when importing more than one file). Required parameter.	
-t	Map type. Possible values: "sdf", "sdfx", "natural", "bms", "mfs" (required parameter).	natural
-s	Server address	127.0.0.1
-m	Indicates where the error line is located: "first", "last", "lastm1" (last minus 1), "lastm2", "lastm3", "lastm4" (Natural maps only)	last
-p	Server port	2323
-u	ApplinX user name (Required parameter)	
-x	The file extension. All files from the given directory that have this extension will be loaded (when not specified, the default extension for the map type is used: ".sdf", ".sdfx", ".ncd" or ".nsm"(for Natural), ".bms", and ".mfs"). When a file name is provided (and not a directory), the specific file will be imported (this parameter is not required in this case).	
-w	ApplinX user password	Empty by default

The screens created appear in the directory you determined in the **Target folder** field. The names of the screens are identical to the map names.

The report is displayed in the command window and includes a list of the screens added as well as the fields and identifiers created/updated/deleted.

Configuring Screens

Define Steps in the Application Map

Refer to the **Application Map** for details about the application map. In this tab it is possible to manually define additional steps between screens that already exist in the application map. The steps should be defined in the source screen and determine the destination screen of the step. It is possible to add a simple step between screens or to add an inner path step which executes a path. When you add a simple step, determine the target screen, the key to be sent, and the relevant inputs. When defining an inner path, select the path. Change the status of the step using the links (Approve, Set to Pending, Set to Not Approved). Once you save the new steps, you can check these navigation definitions in the Session View using the Application Map toolbar. Simply select the screen you want to navigate to and click on the Go icon to attempt to navigate to that screen.

Define Screen Based Tables

A screen based table represents a logical table based on data gathered from a single ApplinX screen. When creating a table you can configure:

- The header: A rectangle that contains all the column headers, and will later be used in the framework to hide this area.
- Filters: empty row, and duplicate row filters will filter the table according to the value defined in the Primary Key column.
- Table Columns: Includes the table definitions such as column details, occurrences, primary keys etc.

Once defined, you may view and design a table in an ApplinX Web application, or access it through a designated API in the ApplinX Base Object or from the Path Procedure Screen Mapper.

To create a table:

1 Access the relevant screen and click on the Tables tab.

2 Click on the **Create a New Table** link.

- A default name (new_table), which you can edit will appear in the Table name field.
- The header definition is filled with values.
- All the multiple fields which are in the screen are added to the list of columns.

3 Ensure that you check and refine the table definitions as required:

- Header: To change the default header definitions, select the header area in the preview of the host screen and then click on the **Capture from screen** icon that is displayed next to the header row and column fields.
- Filter rows: There are two types of filters which can be defined: filtering of empty rows (rows in which the values of the primary key columns are empty) and filtering of duplicated rows (rows in which the values of the primary key columns are identical to those of a previous row). By default these checkboxes are not selected.
- Remove, add or edit table columns:
  - In the Name field enter a logical name to use when programming using the API or the framework, in the Caption field enter the name of the column as it will appear in the Web application, and in the Mapped to field select the host field to which the column will point.
  - Use the Move Up and Move Down options to define the order of the columns.
  - Define the table's primary key. A primary key is a key in the table that uniquely identifies each record. A table must always have one and only one primary key. The primary key can consist of one column or a combination of several columns, whose values give a unique key of each record. If no column is defined as the primary key of the table while creating the table, ApplinX automatically creates a numeric primary key for each record.

    To define a primary key, select the relevant column and click on the **Define Primary Key** link.

Associate Screen Groups

A screen that belongs to a group inherits all the application field mappings that are mapped to the group.

To associate screen groups to a screen

1 Access the relevant screen and click on the Assigned Screen Groups tab. The list of Screen Groups currently associated with this screen are listed.

2 Click on Assign Screen Group to select additional screen groups to be associated with this screen.

Determine Screen Direction (relevant for bi-directional applications)

This tab is only displayed in right-to-left applications and enables overriding application settings per a specific screen.

VT Parameters (relevant for VT hosts only)

VT parameters are defined per application. Sometimes it is necessary to override these parameters for a specific screen. This can be implemented by opening the relevant screen in the Editor view and in the VT parameters tab, changing the configuration. The description of each of these parameters is included in the description of the VT application parameters: VT Application parameters.

Apply Transformations

Note: This is not available when working with a SOA license.

The Transformations tab lists the transformations used in the screen. Next to each transformation, an icon indicates whether the transformation is inherited from a screen group, whether the transformation has been defined locally for the current screen, or whether the transformation was originally inherited from a screen group but has been disabled for the current screen.

The Add Transformation Mapping hyperlink enables associating a predefined transformation with this screen. The Delete Transformation hyperlink removes the transformation from the list.

The Enable/Disable icon allows you to determine whether to use the selected inherited transformation or not. The Edit icon enables defining the region relevant to the transformation. The Add icon enables associating a predefined transformation with this screen. The Delete icon (enabled on when selecting a local transformation) removes the transformation from the list.

To apply a transformation to a screen:

1 In the relevant screen, click on the **Transformations** tab.

2 Click the **Add Transformation Mapping** link to add a transformation.

3 Select the transformation from the drop-down list of transformations or click on the **New** button to create a new transformation.

4 Select the relevant screen area: "Anywhere on the screen", "Application fields" (it is mandatory to select a field once this option has been selected) or "Rectangle" and define the region parameters as necessary.

Previewing a Screen

You can display an HTML preview of the screen. This is particularly useful to preview transformations that are used in this screen.

To preview a screen

- Right-click on the relevant screen and select **Open HTML Preview**.

  Note: The calendar component used in Calendar transformations is not displayed when previewing the screen in the HTML preview.

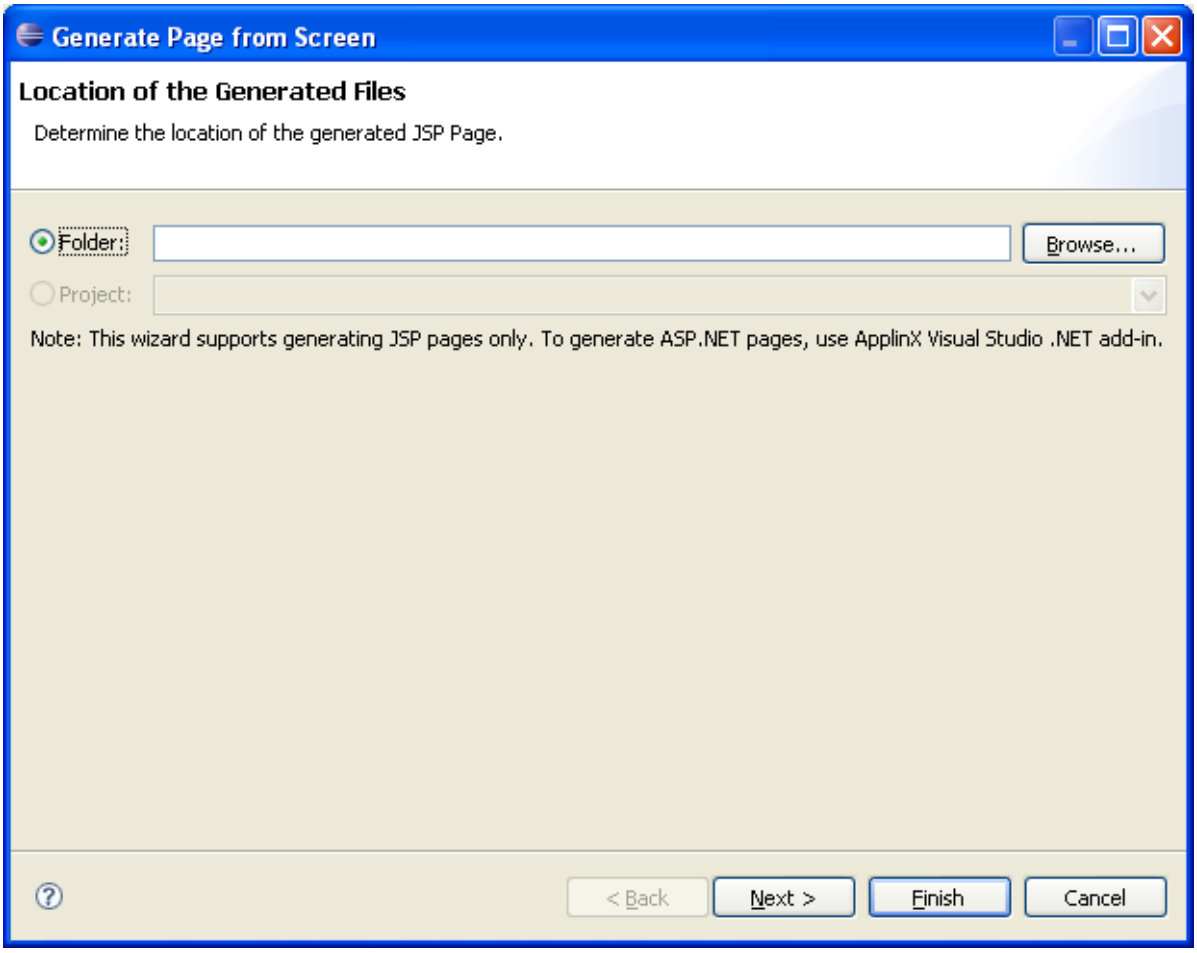
Generating a JSP/ASPX Page from a Screen

When the Instant Web Application, which enables enhancements such as relatively simple visual changes does not meet all the requirements, it is recommended to generate JSP/ASPX framework pages that can be modified, customized or further enhanced. Such required modifications may include using a custom template for a screen or screen group, developing custom design (adding custom images, hyperlinks etc.) or developing custom functionality (unifying screens, handling tables, adding custom Word/excel/ graphic integration etc.).

A JSP/ASPX pages can be generated for a screen or screen group when connected online or when not connected online.

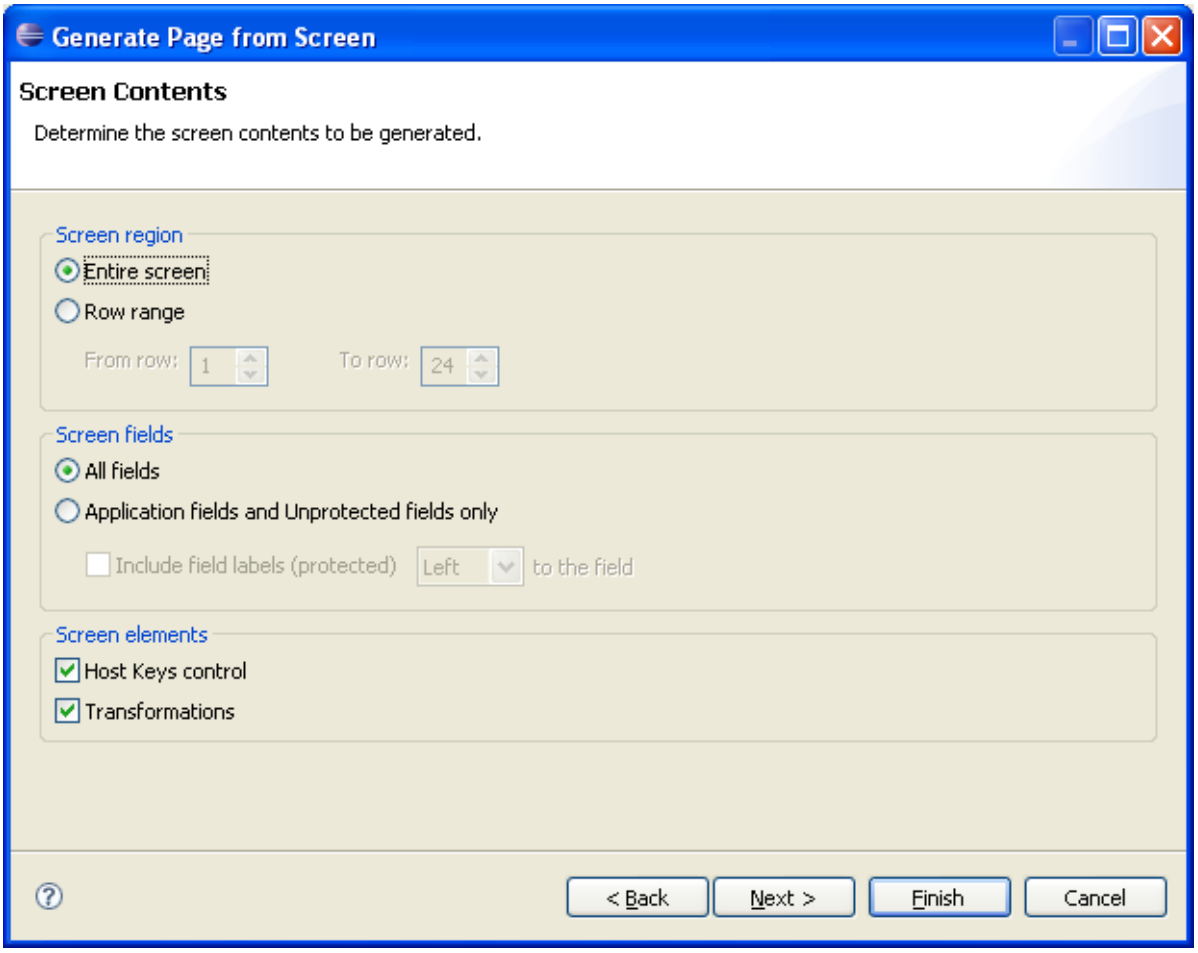
To generate a JSP page:

1 Right-click on the relevant screen in the ApplinX Explorer and select **Generate JSP Page from Screen**.



2 Determine the folder where the generated page will be placed.

3 Click **Next**. The *Screen Contents* screen is displayed.

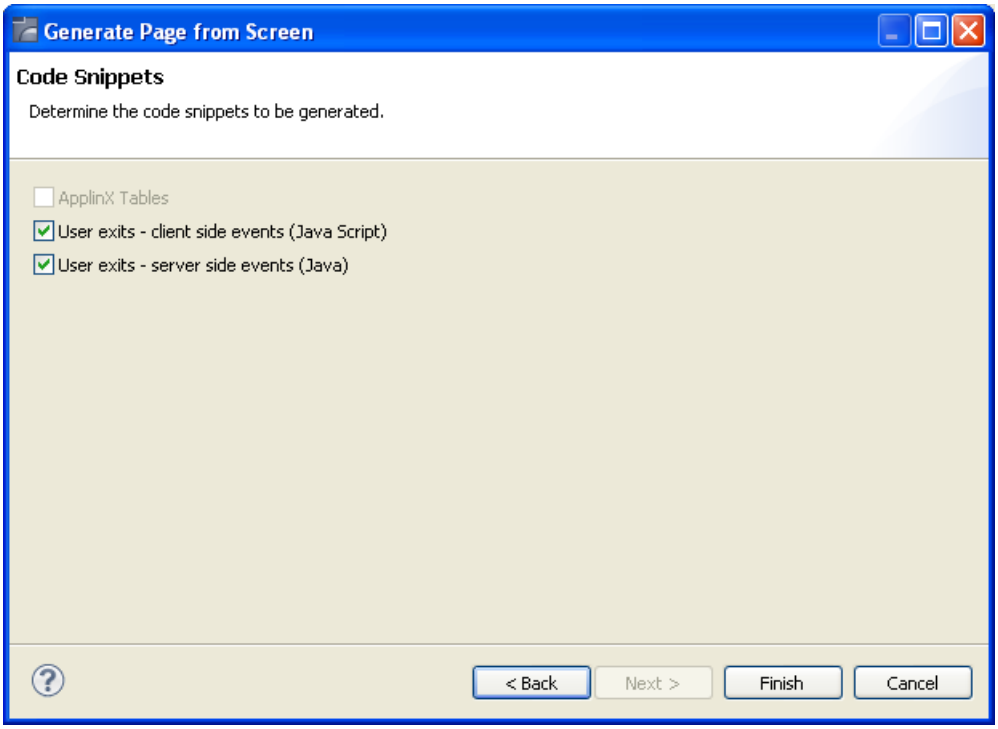


4 Determine the region in the screen that is to be generated: this can be the entire screen or a specific range of rows.

5 Determine the type of fields that are to be generated: all the fields or only application and unprotected fields.

6 Determine whether the host keys control and/or transformations are to be generated.

7 Click **Next**. The *Code Snippets* screen is displayed.



8 In the Code Snippets tab define the elements you would like to generate code behind for. The code class contains methods which can be used as a basis for further development. Click **Finish**.

To generate an ASPX framework page:

- Use the ApplinX Visual Studio .NET add in.

13 Screen Groups

What is a Screen Group?

The Screen Group entity binds several Screens that share common visual or logical attributes. The relationship between Screen Groups to Screens is Many-To-Many: a screen group can include many screens, and a screen may be included in many groups.

A screen that belongs to a group inherits all the application field mappings that are mapped to the group. Unidentified screens are associated with screen groups based on the screen group identifiers defined. Identified screens relate to specific, predefined screen groups (specified in the Screen entity dialog box), ignoring the screen group identifiers. In addition, it is possible to define a screen group that includes all the unidentified screens in the application.

Screen groups are typically used:

- To identify a large number of Host Screens without having to identify all the relevant ApplinX screens.
- To attach a different Web template to different parts of the host application.
- To define once a Field Mapping that is repeated in multiple screens (error message, screen title, command line, menu selection, etc.).
- To perform Web transformations on a set of screens without customizing a Web page for each screen.
- To implement **Screen Creation Definitions**

Note that when there are a number of screen groups, it is necessary to determine the order of priority between the screen groups. For more details refer to **Prioritizing Screen Groups**

After creating a screen/screen group, you can preview how it will look by right-clicking on the screen/screen group and selecting **Open HTML Preview**.

In this section we cover:

Creating a Screen Group

The Screen Group entity binds several Screens that share common visual or logical attributes. Each screen group must have unique name. A screen group may have identifiers enabling the screen group to be implicitly associated to specific unidentified screens or have no identifiers, enabling the screen group to be associated with all unidentified screens.

To create a new Screen Group

1 Select the relevant application.

2 In the **File** menu select **New>Entity>Screen Group**. The *New Screen Group wizard* is displayed.

3 Enter a name for the Screen Group, a suitable description and determine the folder where the screen is to be located. Click **Finish**. You have now created a Screen Group entity in the repository.

Define Identifiers

For the ApplinX Server to recognize a host screen, you need to identify the screen in ApplinX. The screen is identified using "identifiers". Different identifier types include identifying specific text on the screen, the screen cursor position or the field attributes. The ApplinX Server analyzes the current screen and tries to match it to all of the screen identification strings stored in the application repository (database). For a screen to be recognized, all its identifiers must be matched. You may define an unlimited number of identifiers.

Text Identifiers: Use text identifiers to identify a screen by selecting specific text that appears on the screen. Use the Capture from Screen feature to identify a string of text accurately from the host screen and insert this string into the text box.

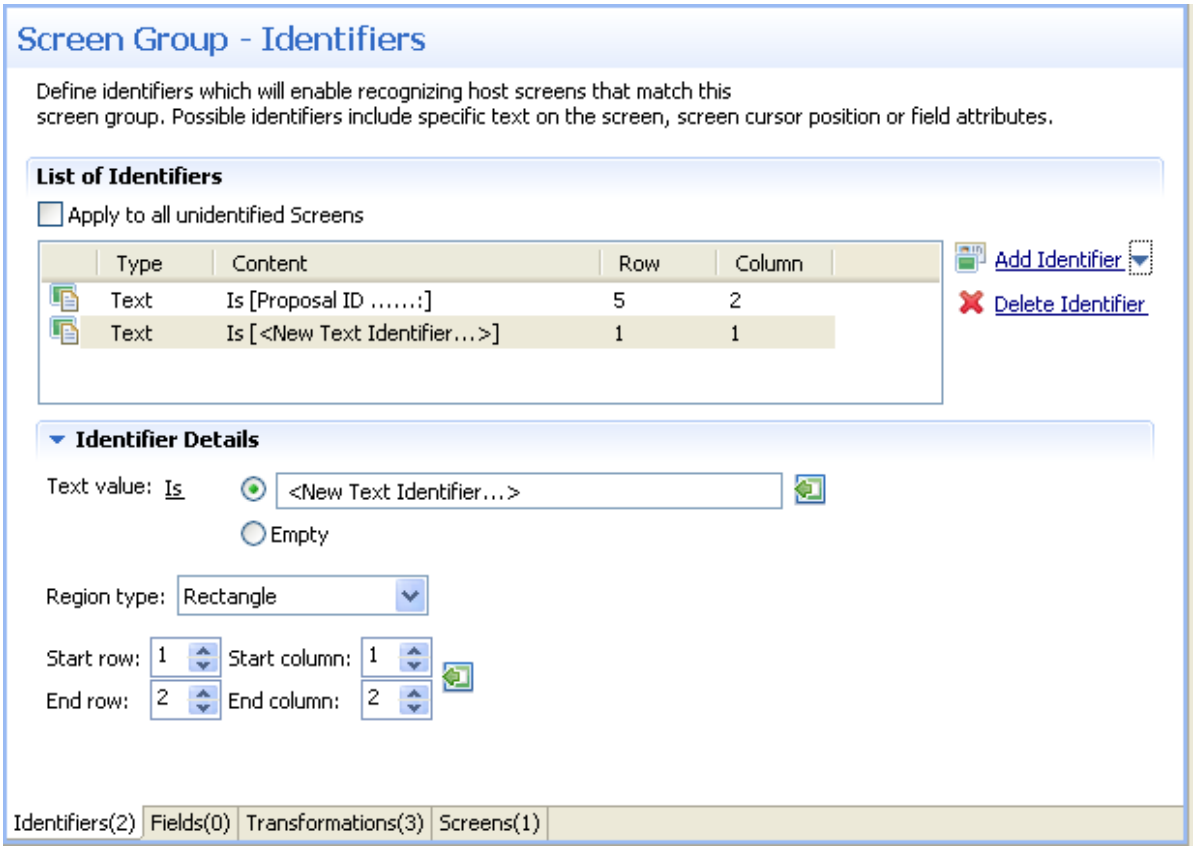
To define a text identifier:

1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 Determine whether to ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

4 Click the arrow on **Add Identifier** and select **Text**. Another row will be added to the list of identifiers.



5 Select **Is** (default) in order to match the exact text on the host screen. Alternatively, select **Is NOT** if any text other than the specified is suitable as a screen identifier. Check **Empty** to identify an empty space on the host screen. This clears the **Text** text box.

6 Select **Rectangle** to indicate that the identifier should be searched for within the defined rectangle, **Position**, to indicate that the identifier must start at a specific position or **Anywhere on screen** to indicate that the identifier may be anywhere on the screen.

7 When selecting Rectangle, define the rectangle range (start and end row and column.). When selecting Position, select the row and column.

8 It is also possible to add an identifier by selecting the area in the host screen, and then clicking Add Identifier. When selecting an area which is a rectangle, identifiers will be added for each and every row of the selected rectangle.

Cursor Position Identifiers: Use the Cursor Position identifier to identify a screen by the cursor position when the screen is loaded.

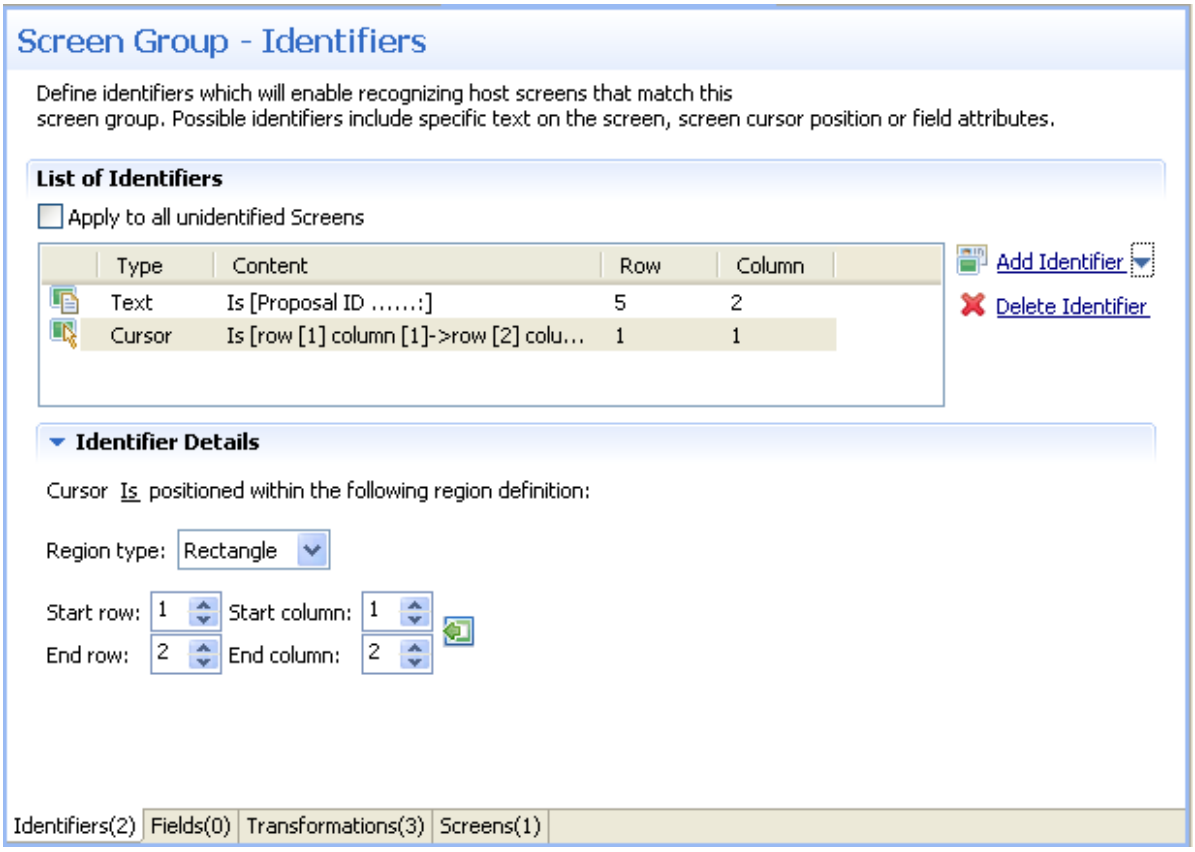
To define a cursor position identifier:

1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 To ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

4 Click the arrow on **Add Identifier** and select **Cursor Position**. Another row will be added to the list of identifiers.



5 Select **Is** to indicate that the cursor is positioned within the region area details that follow. Alternatively, select**Is NOT** to indicate that the cursor is not positioned within the region area details that follow.

6 Select **Rectangle** to indicate that the cursor position should be searched for within the defined rectangle or **Position**, to indicate that the cursor position must start at a specific position

7 It is also possible to add an identifier by selecting the area in the host screen, and then clicking Add Identifier.

Attribute Identifiers: Use the Attributes identifier to identify a screen by the host's attributes. Attributes of the host screen may be Protected, Intensified or Hidden.

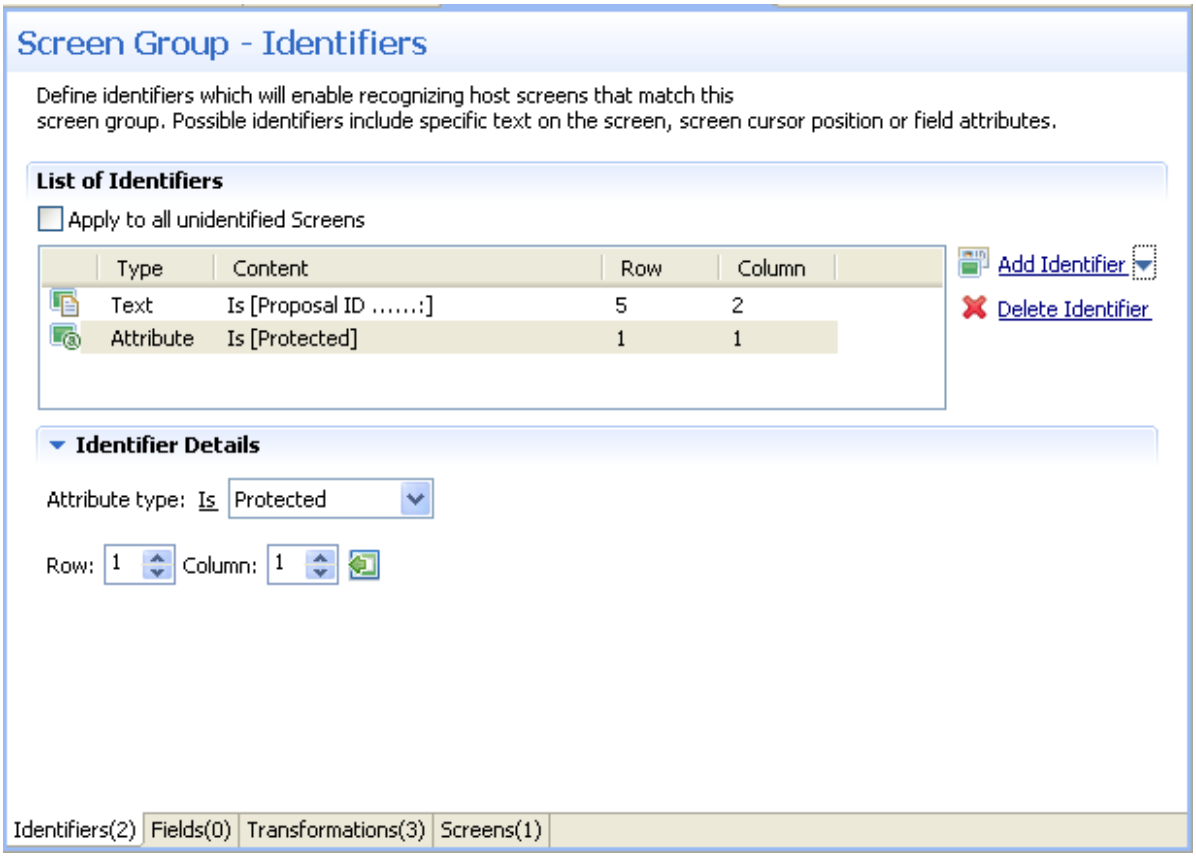
To define an attribute identifier:

1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 To ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

4 Click the arrow on **Add Identifier** and select **Attribute**. Another row will be added to the list of identifiers.



5 Select whether the Attribute Type **is/is not Protected**, **Hidden**, **Intensified** or **Reversed video**.

6 Enter values or use the Capture from Screen feature to indicate a specific location of the attribute by its **Row** and **Column**.

7 It is also possible to add an identifier by selecting the area in the host screen, and then clicking Add Identifier.

Window Identifiers: Use the Window identifier to identify a screen by determining whether it is or is not a window. In Natural UNIX, it is possible to identify a screen by determining whether it is a window and whether it has specific text in the title.

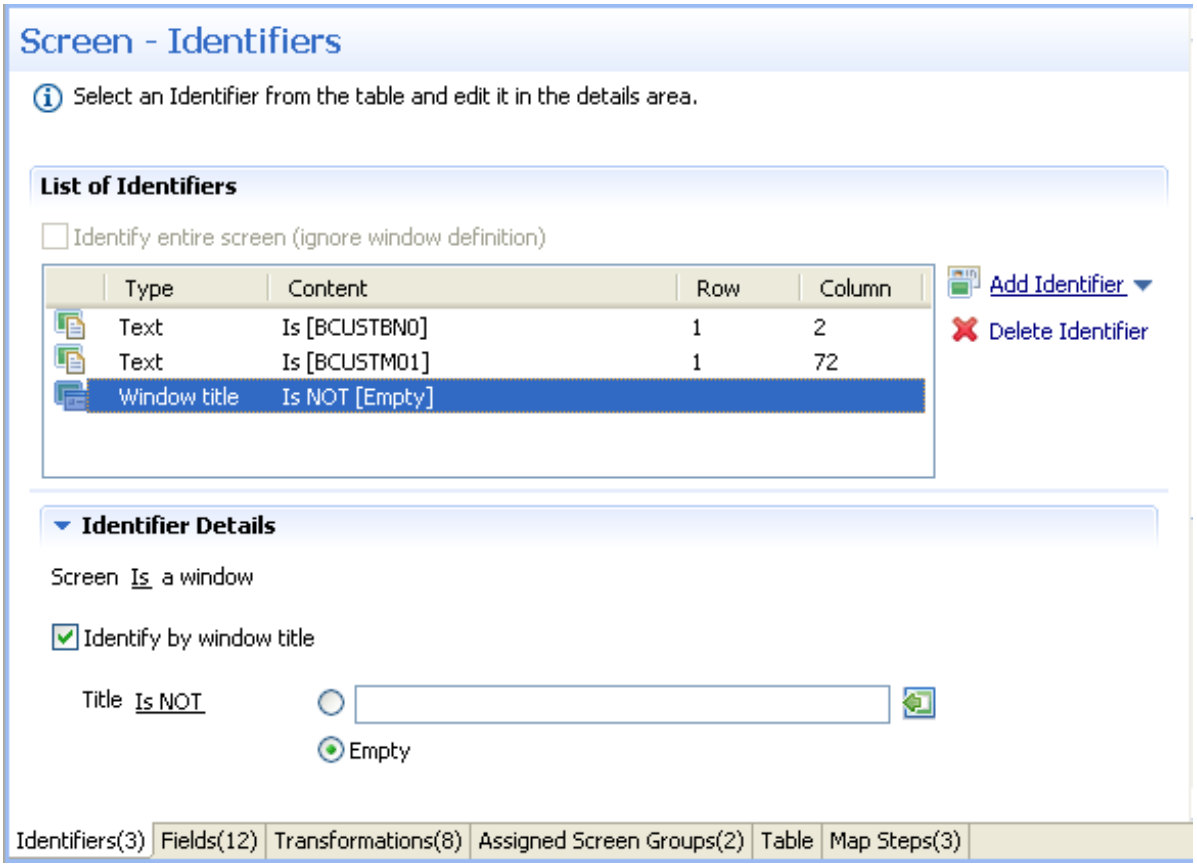
To define a window identifier:

1 Access the relevant screen.

2 Select the **Identifiers** Tab.

3 To ignore the application's window definitions during the identification of a screen, select the relevant check box. Once the screen is identified the full screen is used.

4 Click the arrow on **Add Identifier** and select **Window**. Another row will be added to the list of identifiers.



5 Select **Is** to indicate that the screen is a window. Alternatively, select **Is NOT** to indicate that the screen is not a window.

6 For Natural UNIX hosts: Select the check box to indicate to identify by the window title. You can enter text for the title or select the text in the session view and click the Capture from screen icon. You can also identify a screen which is a window and does not have a title by selecting the Empty radio button.

Map Fields

What is an ApplinX Field?

When a screen is identified, field entities can be mapped to it. A field makes relating to a string on the host screen much simpler. Fields offer an enhanced way of referring to host fields - by their name rather than by their position. In addition, mapped application fields simplify future application maintenance - when a field changes in the original host application, updates only need to be made to the application field definition. The following guidelines will help you determine which fields to identify:

- Fields that will be referenced in code (when binding Web page controls to host fields by name).
- Fields that will be part of navigation paths.
- Fields that will be used as Procedure input and output attributes to expose host transactions as Web services.
- Fields that will be used when creating generated Web tables.
- Recommended: all input and output fields (not constants).
- Any important field in the host application.

An application field makes relating to a string on the host screen much simpler. Fields offer an enhanced way of referencing host fields - by their name rather than by their position. Using fields has many advantages:

- A field's ID is meaningful regarding the contents of the field, unlike a numeric position.
- You define a field only once, and later can reference that definition in all of your code - knowing you always mean the same definition.
- If changes in the host application occur in the future, the only place that requires to be changed is the field definition.

Field properties can be accessed by right-clicking on the Repository node and selecting **Open Entity (Find)...** and then searching for the specific field.

In the Field Editor, it is possible to set the Input Mask (Defines the values that can be entered in the field) and the field type (numeric or alphanumeric).

Input Masks

An input mask consists of literal characters along with special characters that together determine the value that may be entered into input fields. When a user defines a mask on an input field, client side validation will be carried out on the input format. Common usage is in date/time and number or currency fields.

To implement this feature you must change the web application configuration in the configuration editor:

1 Open a new browser and run your Web application.

2 Click on the Configuration link. The Configuration Editor will be displayed.

3 In the **Instant** node select the **Reflect emulation behavior** check box.

The following table shows some useful input mask definitions and examples of values you can enter. Refer to Valid input Characters for details on the codes used to create input mask definitions.

Input Mask Definition	Example Values
(000) 000-0000	(206) 555-0248
(999) 999-9999	(206) 555-0248 ( ) 555-0248
(000) AAA-AAAA	(206) 555-TELE
#999	-20 2000
>L????L?000L0	GREENGR339M3 MAY R 452B7
00000-9999	98115- 98115-3007
L?????????????	Maria pierre
ISBN 0-&&&&&&&&&&-0	ISBN 1-55615-507-7 ISBN 0-13-964262-5

Valid Input Mask Characters

ApplinX interprets characters in the Input Mask property definition as shown in the following table. To define a literal character, enter any character other than those shown in the table, including spaces and symbols. To define one of the following characters as a literal character, precede that character with a "\".

Character	Description
0	Digit (0 through 9, entry required; plus [+] and minus [-] signs not allowed).
9	Digit or space (entry not required; plus and minus signs not allowed).
#	Digit or space (entry not required; blank positions converted to spaces, plus and minus signs allowed).
L	Letter (alphabetic, entry required).
?	Letter (alphabetic, entry optional).
A	Letter or digit (entry required).
a	Letter or digit (entry required).
&	Any character or a space (entry required).
C	Any character or a space (entry optional).
\	Causes the character that follows to be displayed as a literal character. Frequently displayed any of the characters listed in this table as literal characters (for example, \A is displayed as just A).

When a screen is identified, field entities can be mapped to it. A field makes relating to a string on the host screen much simpler. Fields offer an enhanced way of referring to host fields - by their name rather than by their position. In addition, mapped application fields simplify future application maintenance - when a field changes in the original host application, updates only need to be made to the application field definition. The following guidelines will help you determine which fields to identify:

- Fields that will be referenced in code (when binding Web page controls to host fields by name).
- Fields that will be part of Path Procedures.
- Fields that will be used as Procedure input and output attributes to expose host transactions as Web services.
- Fields that will be used within tables.
- Recommended: all input and output fields (not constants).
- Any important field in the host application.

There are limitations when defining a mapping. A mapping cannot:

- Start in an unprotected field and end in a protected field, or vice versa.
- Start with an attribute byte.
- Overlap another mapping on the same screen.
- Start in one line and end on another line.
- Start on one line and end off the screen's limits.

When using Screen Groups, you may want to associate to a specific Screen Group all the screens which contain a certain field, wherever this field may appear on the screen. For example, you may want all screens which contain a menu selection field to belong to the same screen group. This is

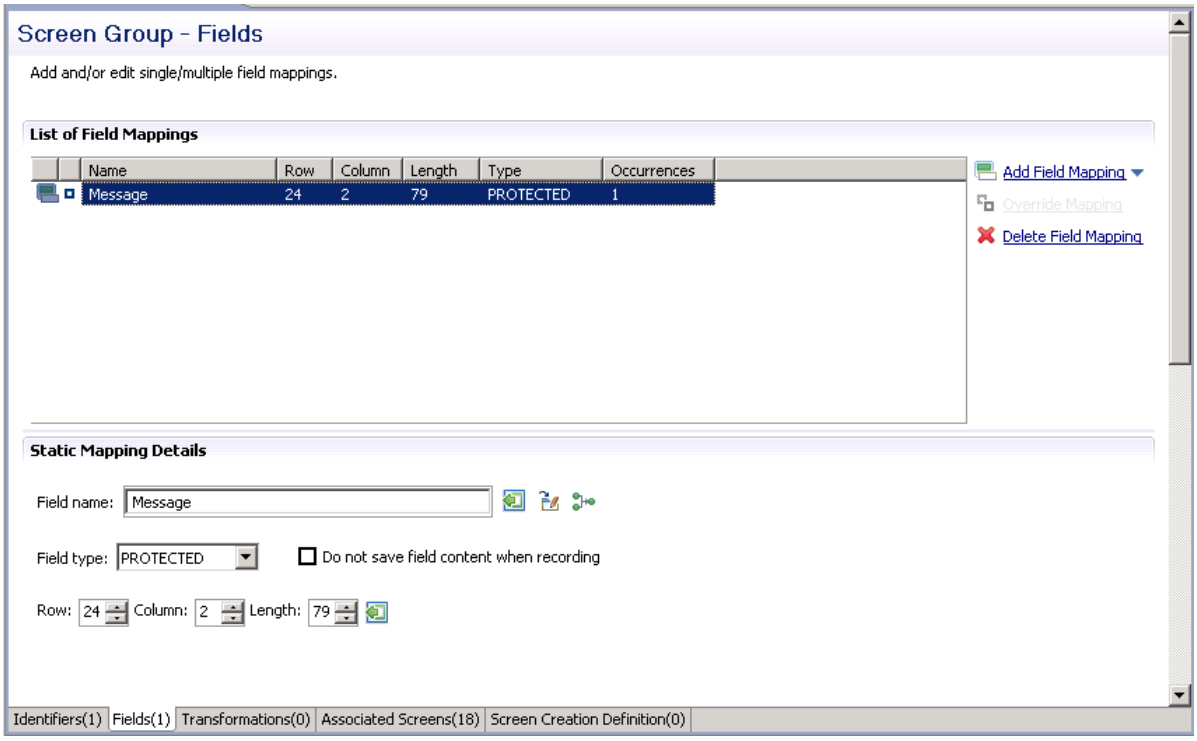
possible using the mapping type called "Single Dynamic". This mapping type maps fields according to their leading label, no matter where this field appears on the screen.

Fields can be mapped according to their position in the screen or according to their leading label:

To map application fields according to their position in the screen:

1 Click the **Fields** tab to view and/or modify the application fields in this screen group.

2 Click the arrow next to **New Mapping**, and select **New Single Mapping** or **New Multiple Mapping** icon to add a new field mapping.



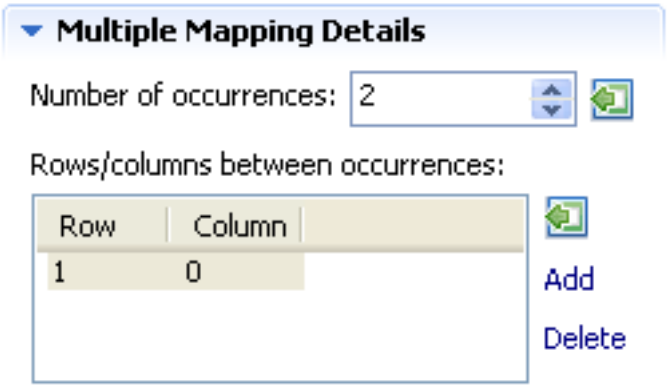
3 In the **Field name** field, either enter a field name or enter the first letters of a field name and then click CTRL and SPACE to display relevant fields.

4 Select the **Protection type**: **Protected** (cannot be edited in host), **Unprotected** (editable in host) or **Both** (may sometimes be editable and sometimes not).

5 The **Do not save field content when recording** check box enables not saving data which you do not want to be displayed when viewing the recorded trace file.

6 The **Row** and **Column** fields indicate the position of the field on the screen. The values of these fields can be changed using the **Capture from Screen** feature or by entering values in the fields.

7 When adding a multiple field mapping, determine the number of occurrences and the rows and/or columns between occurrences.



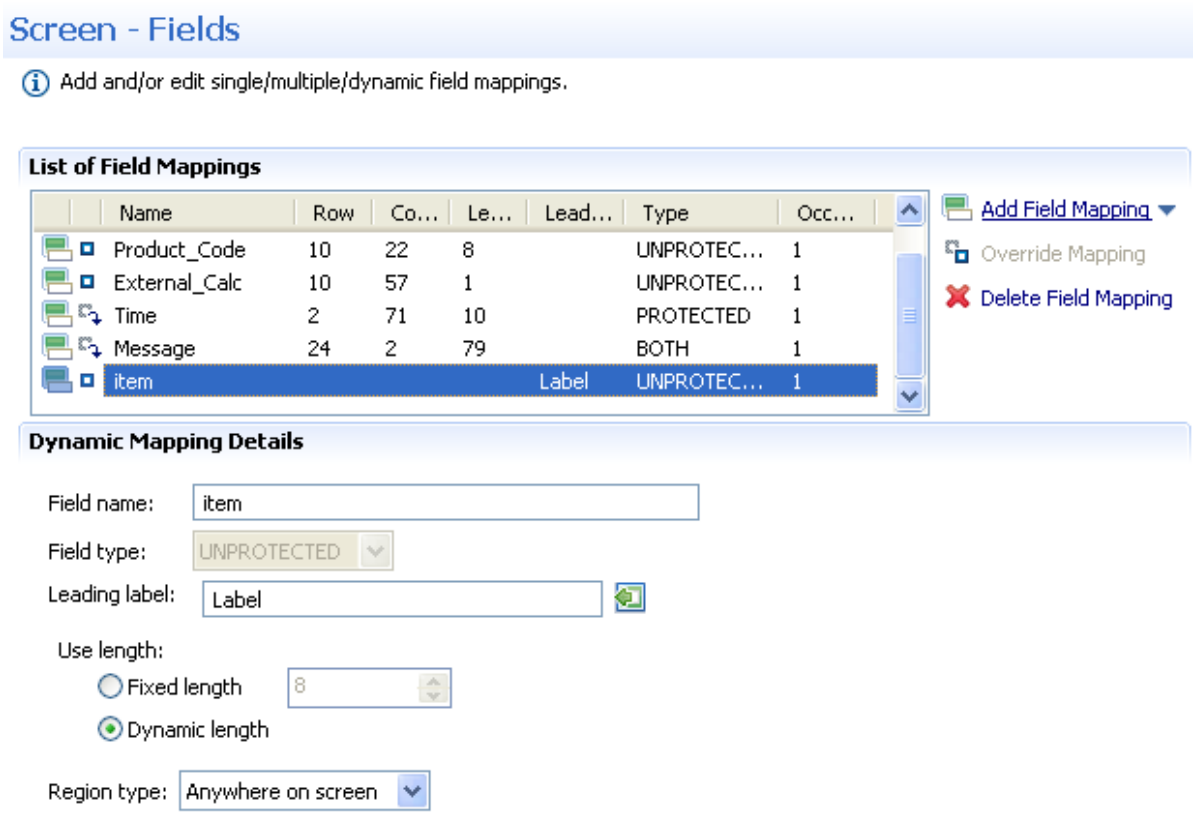
To delete a field mapping, click on the **Delete Field Mapping** hyperlink.

To map application fields according to their leading label (not available in right to left and character mode applications):

In some screens, the position of a field may vary. It is therefore necessary to map some fields in such a way, that even if they appear in a different position, they will still be recognized and mapped. This is possible by defining the label near the field (the label must be to the left of the field) and identifying the field according to this label. This mapping type is called "Single Dynamic" as the mapping position changes dynamically according to the leading label. Use this type of mapping only when the screens in the screen group have a very similar structure. Refer to Dynamic Field Mapping Limitations.

1 Click the **Fields** tab to view and/or modify the application fields in this screen.

2 Click the arrow next to **New Mapping**, and select **Single Dynamic** to add a new field mapping.



3 In the **Field name** field, either enter a field name or enter the first letters of a field name and then click CTRL and SPACE to display relevant fields.

4 Select the field protection type: Unprotected, Protected or Both. Prefer to set the field type either as **Protected** or **Unprotected** when you are sure of the type of the dynamic field. Use **Both** when you are not sure of the type of the dynamic field.

5 Determine the leading label: the leading label is the label which appears before the field. This label is inserted as the Field name and can be edited. The label you define should be as accurate as possible (do not use strings which normally appear within the field).

6 Determine the method to use to search for the label:

- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note: When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

7 Define the region type in which the label will be searched for: anywhere in the screen, within a specified rectangle or within a specified row range. The **Row** and **Column** fields indicate the position of the label on the screen. The values of these fields can be changed using the **Capture from Screen** feature or by entering values in the fields.

8 Use length:

In Unprotected fields: Determine whether to recognize a field mapping according to the leading label and the length of the input field (Fixed length) or just according to the leading label (Dynamic length).

In Protected fields: The length defined here is the length of the mapping. The identified field is cropped to the specified length.

9 Define the region type in which the field will be searched for: anywhere in the screen, within a specified rectangle or within a specified row range. The **Row** and **Column** fields indicate the position of the field on the screen. The values of these fields can be changed using the **Capture from Screen** feature or by entering values in the fields.

Note: The region will be displayed in the Session View with a yellow frame, and the background of the screen within this region will be dotted.

Recommended guidelines for mapping fields according to their leading labels:

- It is preferable to define your regions as rectangles, that do not overlap each other, and whose size are minimal.
- The feature is best suited to locate dynamic fields which change their location vertically so that labels are aligned vertically in one region, and fields aligned vertically on a region to the right of the labels.

Configuring Screen Groups

- Converting a Screen to a Screen Group/Converting a Screen Group to a Screen

Associate Screens

A screen that belongs to a group inherits all the application field mappings that are mapped to the group.

To associate screens to a screen group

1 Access the relevant screen and click on the Associated Screens tab. The list of Screen Groups currently associated with this screen are listed.

2 Click on Assign Screens to select additional screens to be associated with this screen group.

Create Screen Creation Definitions

Screen Creation Definitions are a set of basic definitions which are used to create a new ApplinX screen. These definitions include the default screen name, initial identifiers and determine whether input fields will be created. Creating a new screen using screen creation definitions reduces the complexity and the time required to identify new screens.

How does it work?

To use this feature, at least one set of Screen Creation Definitions must be created within a Screen Group (in the dedicated tab). When navigating within a session, and reaching an unidentified screen, ApplinX searches for Screen Groups which match the unidentified screens, and then creates a new screen based on the screen group's identifiers and on parameters defined in the creation definition configured in the Screen Group. This process can be initiated in three different modes: automatic, semi-automatic and manual (defined in the view session). Refer to Changing the Screen Definition Mode.

To create screen creation definitions

1 In the Screen Creation Definition tab, click on Create Definition to define a new screen creation definition.

2 Determine the position of the screen name in the host screen. Click on the Capture from screen icon after selecting the relevant position in the screen to automatically enter the position.

3 Determine the location of the identifiers to be used in the new screen. It is possible to add a location by selecting the area in the host screen, and then clicking Add Identifier Location.

4 Determine whether to automatically create input fields.

5 When automatically creating input fields, determine whether to suggest nearby labels as field names and whether to add the screen name as a prefix to the input fields' names.

Apply Transformations

Note: This is not available when working with a SOA license.

The Transformations tab lists the transformations used in the screen group. Next to each transformation, an icon indicates whether the transformation has been defined locally for the current screen group, or whether the transformation has been disabled for the current screen group.

The Add Transformation Mapping hyperlink enables associating a predefined transformation with this screen. The Delete Transformation hyperlink removes the transformation from the list.

To apply a transformation to a screen:

1. In the relevant screen, click on the **Transformations** tab.
2. Click the **Add Transformation Mapping** link to add a transformation.
3. Select the transformation from the drop-down list of transformations or click on the **New** button to create a new transformation.
4. Select the relevant screen area: "Anywhere on the screen", "Application fields" (it is mandatory to select a field once this option has been selected) or "Rectangle" and define the region parameters as necessary.

Prioritizing Screen Groups

It is necessary to determine the order of priority of the screen groups in the following cases:

- When two screen groups match an unknown screen and contain the same field but with different mapping position of the field. The order of the screen groups will determine which screen will be used.
- When using transformations, the transformations of the screen group with a higher priority will be applied to the current screen before the transformations of a screen group with a lower priority.
- In the framework, when generating a page for two screen groups and when two screen groups match an unknown screen, the screen group appearing former in the list will be applied.

To prioritize screen groups

1. In the repository, right-click on any Screen Group and select **Prioritize Screen Groups...**. The **Prioritize Screen Groups** dialog box is displayed.
2. Determine the order of precedence of the screen groups by clicking on the **Move Up** and **Move Down** links.
3. Click **OK** to save your changes.

Previewing a Screen Group

You can display an HTML preview of the screen group. This is particularly useful to preview transformations that are used in this screen group.

To preview a screen group

- Right-click on the relevant screen and select **Open HTML Preview**.

  Note: The calendar component used in Calendar transformations is not displayed when previewing the screen in the HTML preview.

Converting a Screen to a Screen Group/Converting a Screen Group to a Screen

Sometimes, after creating a screen, you may notice that it is relevant to a number of screens and therefore would like it to be a Screen Group. On the other hand, you may have created a screen group which you later realize is suitable to be a screen. ApplinX provides the option to convert a screen to a screen group and vise versa simply by right-clicking on the relevant screen/screen group and selecting Convert to Screen Group/Convert to Screen.

14 Application Map

The Application Map view displays thumbnails of the application screens which the user navigated through, while working with the application. ApplinX saves the navigation through these screens including the steps between each screen such as the action key pressed and the fields sent. The application map can be used in Path Procedures and from the Base Object/Web framework, to navigate to a specific screen, using the NavigateTo method.

This section details:

Creating Steps

Steps can be created automatically from within the Session View when working online or offline, and also via a wizard, using a number of trace files. Steps can also be added manually in the Steps tab of the Screen editor. To automatically add steps as you navigate, you must first enable recording the navigation steps. To perform the identification of steps based on a trace file, use the Automatically Identifying Application Map Steps wizard, and all the steps in the trace file will be identified. Each of these tasks are detailed below.

Note: Steps can only be added to identified screens.

Note: When working with an online session, the steps (i.e action keys pressed) between each screen, are the actual keys you press. When working with a trace file, the steps (i.e action keys pressed) between each screen, are the steps recorded within the trace file.

- Recording Navigation Steps from within the Session View
- Automatically Identifying Application Map Steps using Trace Files (GCT)
- Manually Identifying Application Map Steps

Recording Navigation Steps from within the Session View

To Record Navigation Steps

1 Recording navigation steps is enabled by default. To change this setting, right-click on an application and select **Properties**.

In the **Navigation** tab, select the check box which enables/disables recording the navigation steps.

2 Connect to a session and start navigating. ApplinX will automatically add the new steps.

Note: When running an offline session, you can just navigate by pressing the ENTER key. The actual steps which were recorded are added.

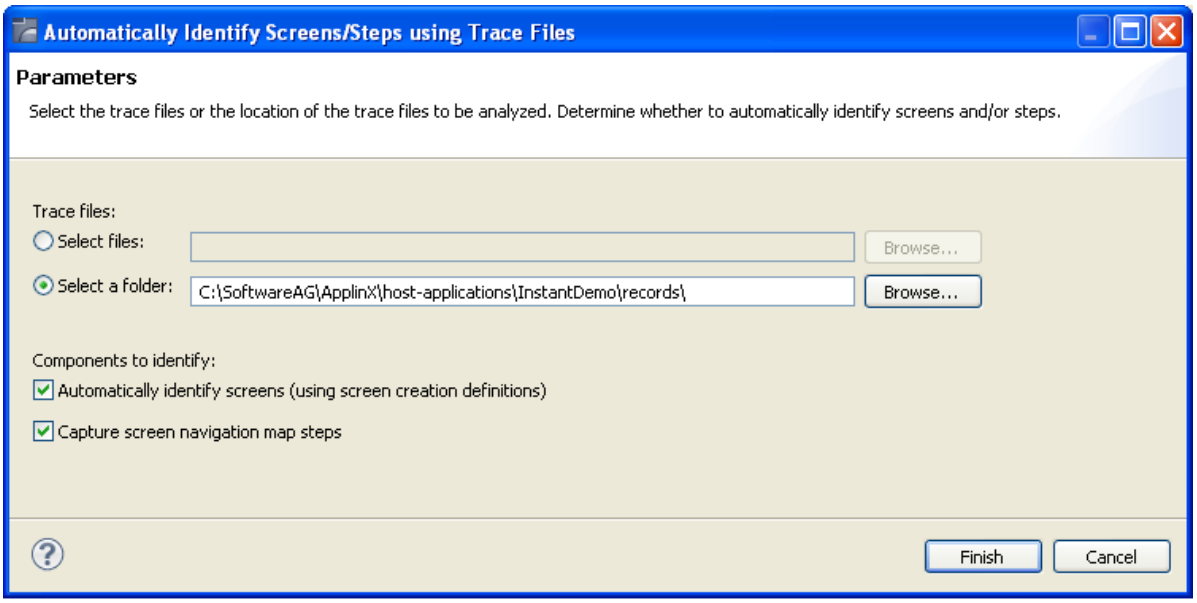
Automatically Identifying Application Map Steps using Trace Files (GCT)

Automatically identify steps by running the *Automatically Identify Screens/Steps using Trace Files* wizard.

Note: This wizard, is also used to automatically identify screens. If your application includes unknown screens which you would like to be displayed in the map, you must identify these screens either manually, or automatically (using this wizard), after first creating a Screen Creation Definition within a Screen Group.

To automatically identify steps using trace files:

1 Right-click on the relevant application, and select **Automatically Identify Screens/Steps using Trace Files...**. The wizard is displayed.



2 Select whether to analyze files or a folder. Locate the files/folder.

3 Select the **Capture screen navigation map steps** check box.

You can automatically identify screens using this wizard by selecting **Automatically identify screens**. Refer to **Automatically Identify Screens using Trace Files (GCT)** for further details.

4 Click **Finish**.

The Console area in the Designer displays the outcome of the process.

Manually Identifying Application Map Steps

In the **Steps** tab of the **Screen** editor, it is possible to manually define additional steps between screens that already exist in the application map. The steps should be defined in the source screen and determine the destination screen of the step. It is possible to add a simple step between screens or to add an inner path step which executes a path. When you add a simple step, determine the target screen, the key to be sent, and the relevant inputs. When defining an inner path, select the path. Change the status of the step using the links (Approve, Set to Pending, Set to Not Approved). Once you save the new steps, you can check these navigation definitions in the Session View using the Application Map toolbar. Simply select the screen you want to navigate to and click on the Go icon to attempt to navigate to that screen.

The Application Map View

Within the Application Map view, you can perform a number of actions:

View Step Information

Display a tooltip with the step information including the key pressed and the information sent. Just hover over the arrow and the tooltip is displayed.

Zoom-in

Change the zoom level of the view. Right click anywhere within the map and select the view zoom.

When moving the mouse over a specific screen image (and the current display is not 100%), the screen image will be enlarged.

Viewing Steps of a Specific Status

The Application Map view displays the steps in different colors to indicate the status of each step: green indicates approved, yellow - pending and red - not approved.

Clicking on the Approved, Pending and Not Approved icons in the toolbar, it is possible to display only the steps of the selected status.

Changing the Steps' Status

Right click on the desired step/arrow and select a status to make changes to the step's status. Using the Control key, select a number of steps/arrows and make status changes to all of these steps.

Link with Entity

The Application Map can either display the whole application map or focus on the entity selected in the Explorer tree and only display the screens related to the selected entity. The radius of related screens can be determined by scrolling to the relevant number in the toolbar. Enable this feature by clicking on the Link with Entity icon, and then selecting the radius of screens to be displayed by clicking on the arrows, or entering a number 

.

Lock/Unlock view

The view displaying the map can be set to change dynamically, as events occur, (e.g., newly created screens are displayed in the map view) or it can be set to preserve the current view (Lock icon on the session toolbar).

Approving Steps

In the Session view, navigate through the screens. In the Application Map view, thumbnails are created for all the identified screens through which you have navigated. Arrows between the screens indicate the direction of the navigation. These arrows are, by default, in "Pending" status (colored in yellow). This is because, you do not always want users to enable users to navigate to all the screens (i.e. you do not want users to be able to navigate to the login page where usernames and passwords appear). In order to use the application map with a Path Procedure, the status of the navigation route must be set to "Approved" (green).You can choose to display or hide all screens that have steps of a certain status (pending, approved or not approved) by clicking on the appropriate icons in the map view.

- To manually add additional steps between screens, edit the source screen in the Editor. In the Map Steps tab manually add steps. Refer to **Creating a Screen**.
- To approve a number of pending steps, right-click on one of the screens and select **Set Pending Steps to Approved**, or use the Control key to select a number of steps.
- You can select to set a screen as the first screen.

Testing the Application Map

The screen navigation defined in the Application Map can be tested to ensure that the navigation behavior is as expected. This is done in the Session View (online), using the Application Map toolbar. The toolbar enables selecting a screen to which you expect to be able to navigate to from the current screen and then attempting to navigate to this screen. If the Application Map is correctly defined, then ApplinX will successfully navigate to the selected screen. If the Application Map does not have the relevant steps defined to reach the screen you selected, a pop-up message will inform you of this.

Troubleshooting

An individual screen thumbnail within the Application Map can indicate problems or incompatibility between the defined identifiers and the screen image attached to the screen. Such problems are indicated by a red frame around the specific screen.

Possible problems:

- Incompatibility between the defined identifiers and the screen image attached to the screen. This can happen as a result of creating a screen and then manually changing an identifier.
- The screen suits the screen image, but there is a screen which is more suitable or identical to the screen image. Look at the name of the screen which is on the screen image and check that it suits the current screen. Note that the name which is on the screen image indicates how the server identified this screen image given all current screen definitions.

15 Transformations

What is an ApplinX Transformation?

An ApplinX transformation contains the definitions required for transforming a certain pattern in the host screen into a Web element. For example, a line pattern in the host screen may be transformed into an HTML Web line.

A transformation can be applied to a Screen Group or to a Screen. Transformations minimize the need to write custom Web code in order to modify the appearance and behavior of ApplinX Web pages.

Refer to Code Transformations for more information regarding the concept of Instant Transformations.

ApplinX Transformations provide many options for transforming host patterns into Web elements. However, it may be the case that it will not be possible to define certain required transformations using the Transformations wizards, as their flexibility is limited. In such cases, it is possible to complement the wizard-defined Transformation entities with code-defined Transformation classes. For more information, see Instant Pages Customization, Code Transformations.

Refer to the Instant demo application and to the self-training files included in the ApplinX installation to see examples of transformations.

Using ApplinX Predefined Transformations

ApplinX predefined built-in transformations provide you a basic kit of commonly used transformations. You can import the predefined transformation entities to help start building up your own transformation library. Once you have imported the transformations, you can use them as-is or customize them to suit your exact needs.

To use ApplinX predefined built-in transformations

- Right-click on the Repository node of the selected application and choose **Import Predefined Transformations**.

Creating a Transformation

To create a transformation:

1 Select the relevant application or the Root node of the application.

2 In the ApplinX Explorer tool bar, click on the arrow to the right of the **Create new entity** icon and select **New Transformation**. The New Transformation wizard is displayed.

3 Enter a name for the transformation, a suitable description and determine the folder where the transformation is to be located. Click **Next**.

4 In the Transformation Outline screen select the initial host pattern and the replacing element. screen. The available Host Pattern Elements:

- Repeating Characters - a repeating sequence of text.
- Text - a textual pattern.
- Input field - an unprotected (input) field.
- Input field with values - an unprotected (input) field followed by a list of its possible values that appears on the same row.
- Date input field - an unprotected (input) field that contains date data.
- Menu - a structured list of options. Each option is preceded by the option code that is placed in a selection input field.

The host pattern elements can be transformed to one of the following application components. The list of application components which appears on the right frame of the Layout screen changes according to the selected host pattern element:

- Line - a horizontal line.
- Text - textual string.
- Hyperlinks - a hyperlink. When relating to menus, each link represents a menu option and its corresponding option code.
- Image - an image loaded from a file.
- Button - a button performing an action when clicked.
- Text field - an input field.
- Combo box - a drop down list with single selection.
- Radio buttons - a list of single-selection buttons.
- Check box - a selected / unselected input element.
- Calendar - a trigger element (image, button or link) that opens a Calendar window.
- Hide - Does not create an alternative component.

5 Click **Finish**. You have now created a new transformation entity in the repository.

For details regarding creating a specific transformation, refer to:

Transforming a Repeating Characters Pattern to a Line

Transforming a Text Pattern to Text

Transforming a Text Pattern to Hyperlink

Transforming a Text Pattern to an Image

Transforming a Text Pattern to a Button

Transforming an Input Field to a Text Field

Transforming an Input Field to a Combo Box

Transforming an Input Field to Radio Buttons

Transforming an Input Field to a Check Box

Transforming an Input Field with Values to a Combo Box

Transforming an Input Field with Values to Radio Buttons

Transforming a Date Input Field to a Calendar

Transforming a Menu to Hyperlinks

Transforming a Repeating Characters Pattern to a Line

This transformation is used to replace repeating characters with a line. It can also be used when you need to remove repeating patterns from text without adding a line (for example, removing unnecessary hyphens or stars, without adding a line). This is done by setting the line element's length to 0.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a Repeating Characters Pattern to a line:

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern.

2 In the **Transform Repeating Characters** tab in the Editor, describe the pattern that is to be searched for in the original host application. Select whether to locate a **Repeating characters pattern** or a more complex (**Advanced**) pattern.

When selecting a Repeating characters pattern, enter the **Repeated characters** and the minimum times that this string is repeated (**Minimum subsequent occurrences**). Select the **Contains title** check box if text may appear between the repeated string. The Preview area will display a preview of the pattern definition.

When selecting **Advanced** pattern, either type in the pattern or use the Capture from Screen feature to directly take a pattern from the host screen. Select the search method to use when searching for this text:

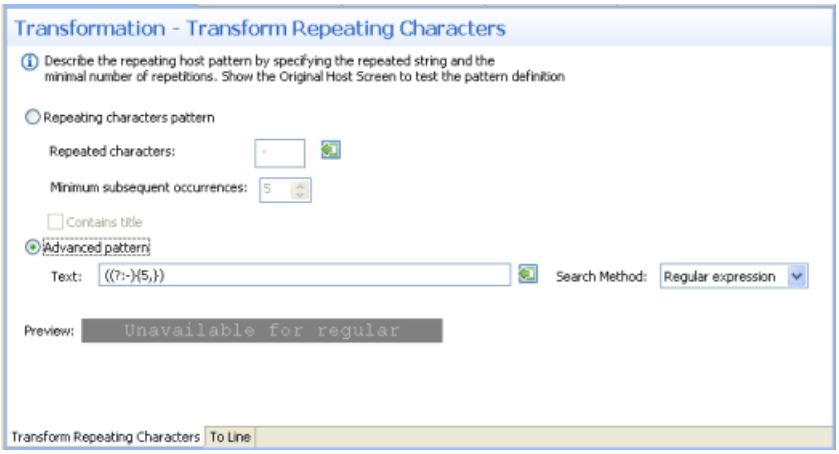
- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note:

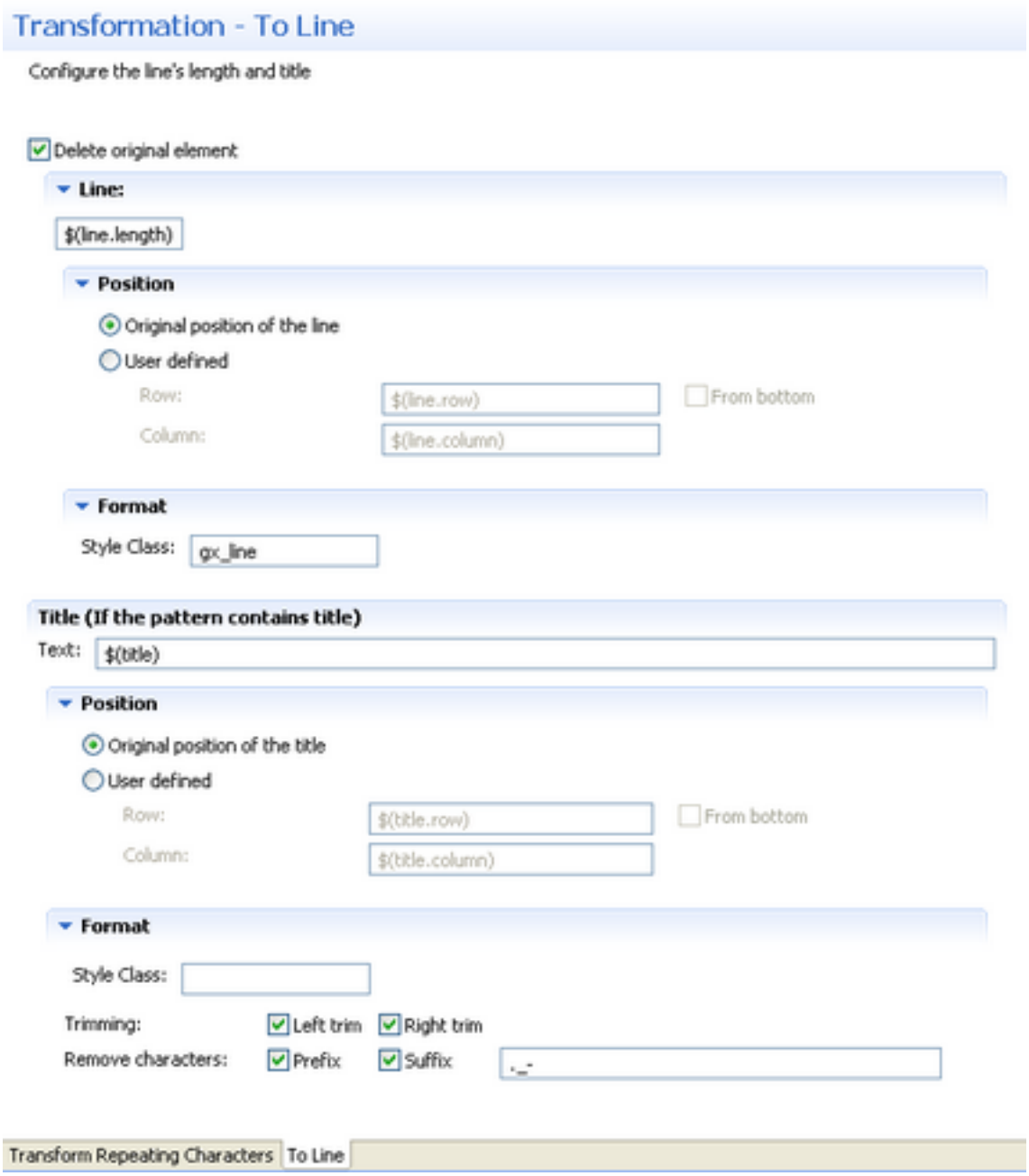
When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

Note: You can view and check the pattern definition by looking at the screen image displayed.



3 In the **To Line** tab in the Editor:



1. Select the **Delete original element** check box to delete the host pattern that was located.
2. Define the length, position and format of the line:
   - Enter the length of the line. By default the length is the length of the original pattern, represented by a variable - `$(line.length)`. Refer to Transformation Variables and their Attributes for further details about setting variables.

- Define the position of the line. By default the position is the original position of the repeating character pattern. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the original pattern: `$(line.row)` and `$(line.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
- Define the format of the line. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the line element.

3. If the original host component includes a title (and you selected **Contains title** in the previous screen), define the line element's title text, position and format:
   - Enter the title text. By default the title name is the original title text and is represented by a variable - `$(title)`.
   - Define the position of the title. By default the position is the original position of the line. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the original pattern: `$(title.row)` and `$(title.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
   - Define the format of the title. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the line element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming a Text Pattern to Text

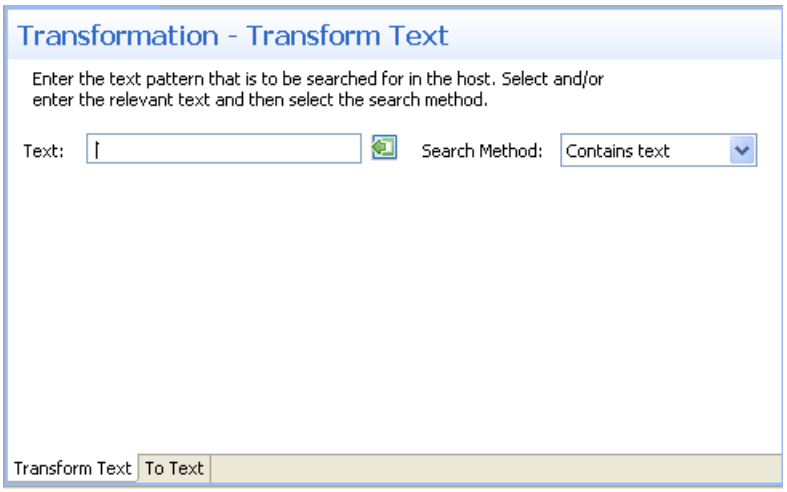
This transformation can not only be used to replace text with other text, but also to format a certain text (for example - remove unwanted characters), leaving the textual data as is, or to move text from one position to another. The latter two usages are implemented by leaving the new text as the default (in the **To Text** tab in the Editor, in the **Text** field, the default value is `$(text)`), preserving the original textual data, and then performing other manipulations such as changing the position, removing unwanted characters or applying a certain style.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a host text pattern to text

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Text** Tab in the Editor, enter the text pattern that is to be searched for in the host. Either type in the pattern or use the Capture from Screen feature to directly take a pattern from the host screen. Select the search method to use when searching for this text:

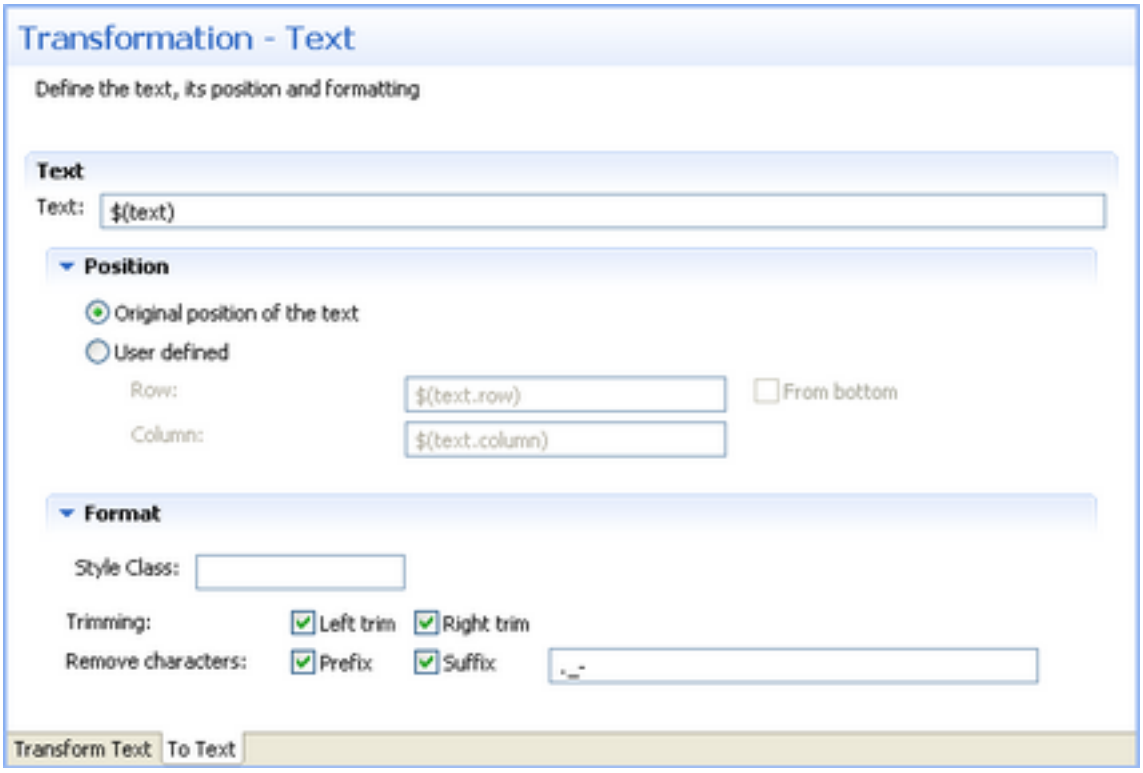
- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note:

When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

3 In the **To Text** Tab in the Editor:



1. Enter the new text. By default the text is the original text and is represented by a variable - `$(text)`. Refer to Transformation Variables and their Attributes for further details about setting variables.
2. Define the position of the text. By default the position is the original position of the text. Select the User defined radio button to define the row and column (by default these values are the row and column of the original pattern: `$(text.row)` and `$(text.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
3. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming a Text Pattern to Hyperlink

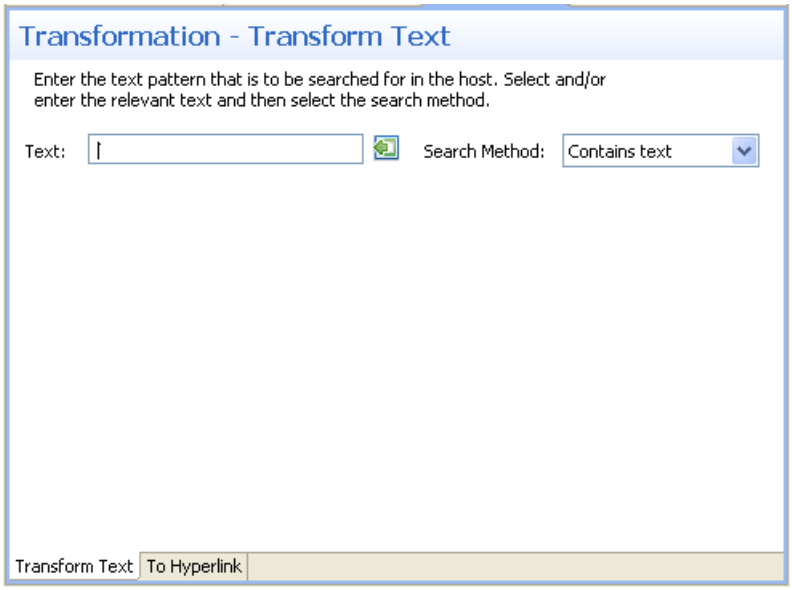
This transformation enables you to transfer a text pattern to a hyperlink. Note that the hyperlink element is a textual element. If you'd like to create a link with an image, refer to **Transforming a Text Pattern to an Image** and select an action that will be performed when the image is clicked (implementing the link behavior).

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a text pattern to a hyperlink

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Text** Tab in the Editor, enter the text pattern that is to be searched for in the host. Either type in the pattern or use the Capture from Screen feature to directly take a pattern from the host screen. Select the search method to use when searching for this text:

- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.

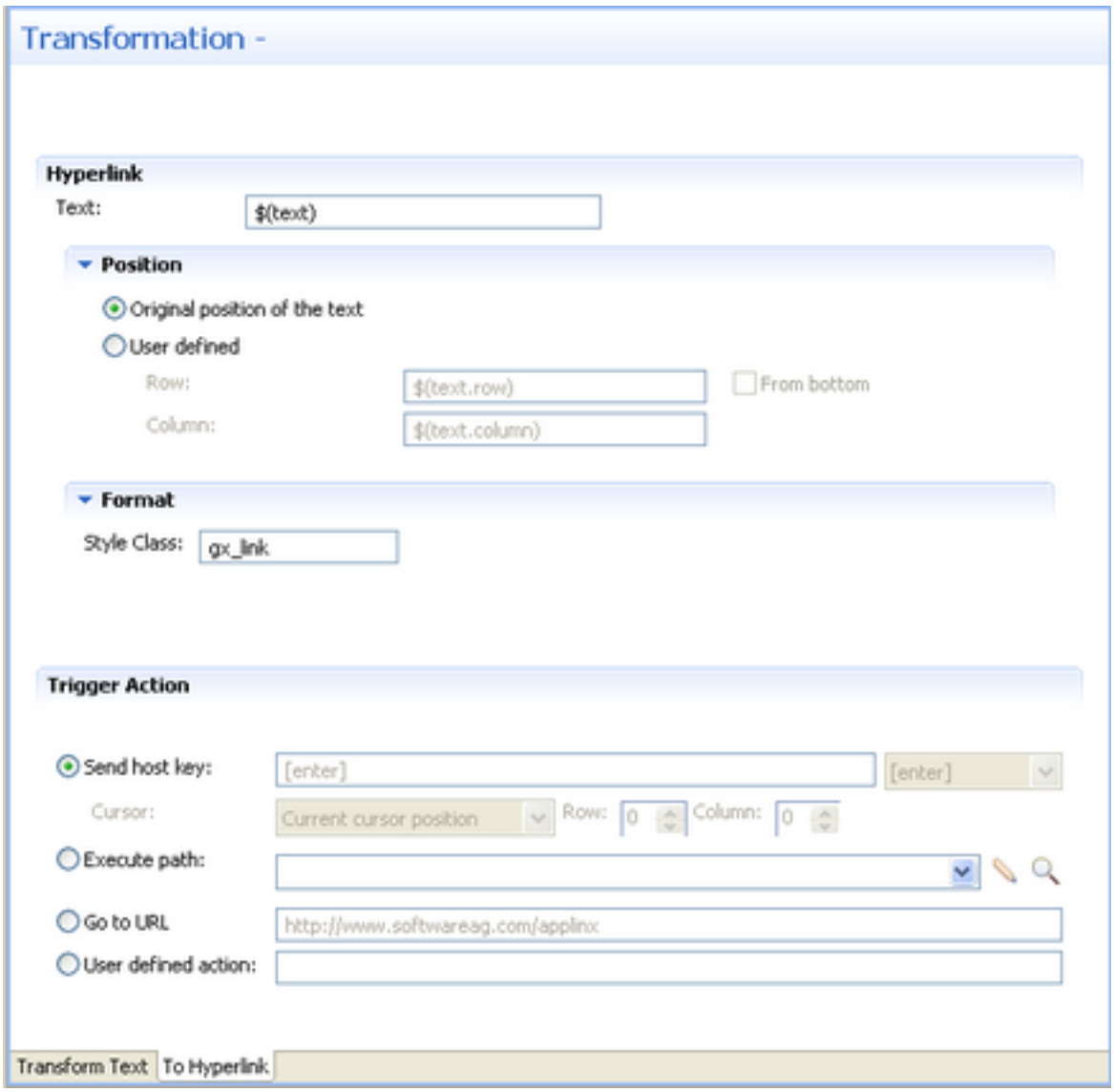
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note:

When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

3 In the **To Hyperlink** Tab in the Editor:



1. Enter the new text. By default the text is the original text and is represented by a variable - `$(text)`. Refer to Transformation Variables and their Attributes for further details about setting variables.
2. Define the position of the text. By default the position is the original position of the text. Select the User defined radio button to define the row and column (by default these values are the row and column of the original pattern: `$(text.row)` and `$(text.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
3. Define the format of the hyperlink. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element.
4. In the Trigger Action section define the action that is going to be performed once the hyperlink is clicked. It is possible to:
   - Send a host key - Select or enter the host key to be sent.
   - Execute a path - Browse to locate the folder where the path is saved. Select the path from the drop-down list.
   - Go to URL - Type in the relevant URL.
   - Enter a User defined action - Type in customized code to perform a customized action.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming a Text Pattern to an Image

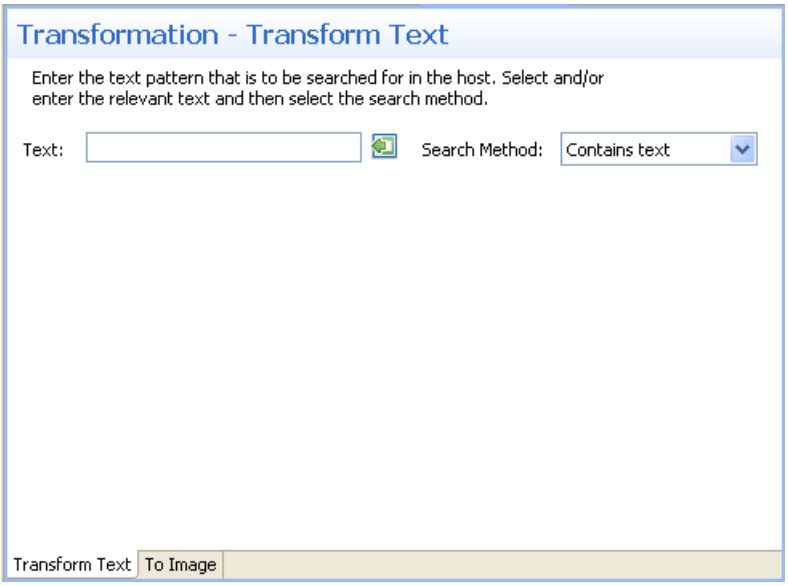
This transformation enables you to transfer a text pattern to an image.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a text pattern to an image

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Text** Tab in the Editor, enter the text pattern that is to be searched for in the host. Either type in the pattern or use the Capture from Screen feature to directly take a pattern from the host screen. Select the search method to use when searching for this text:

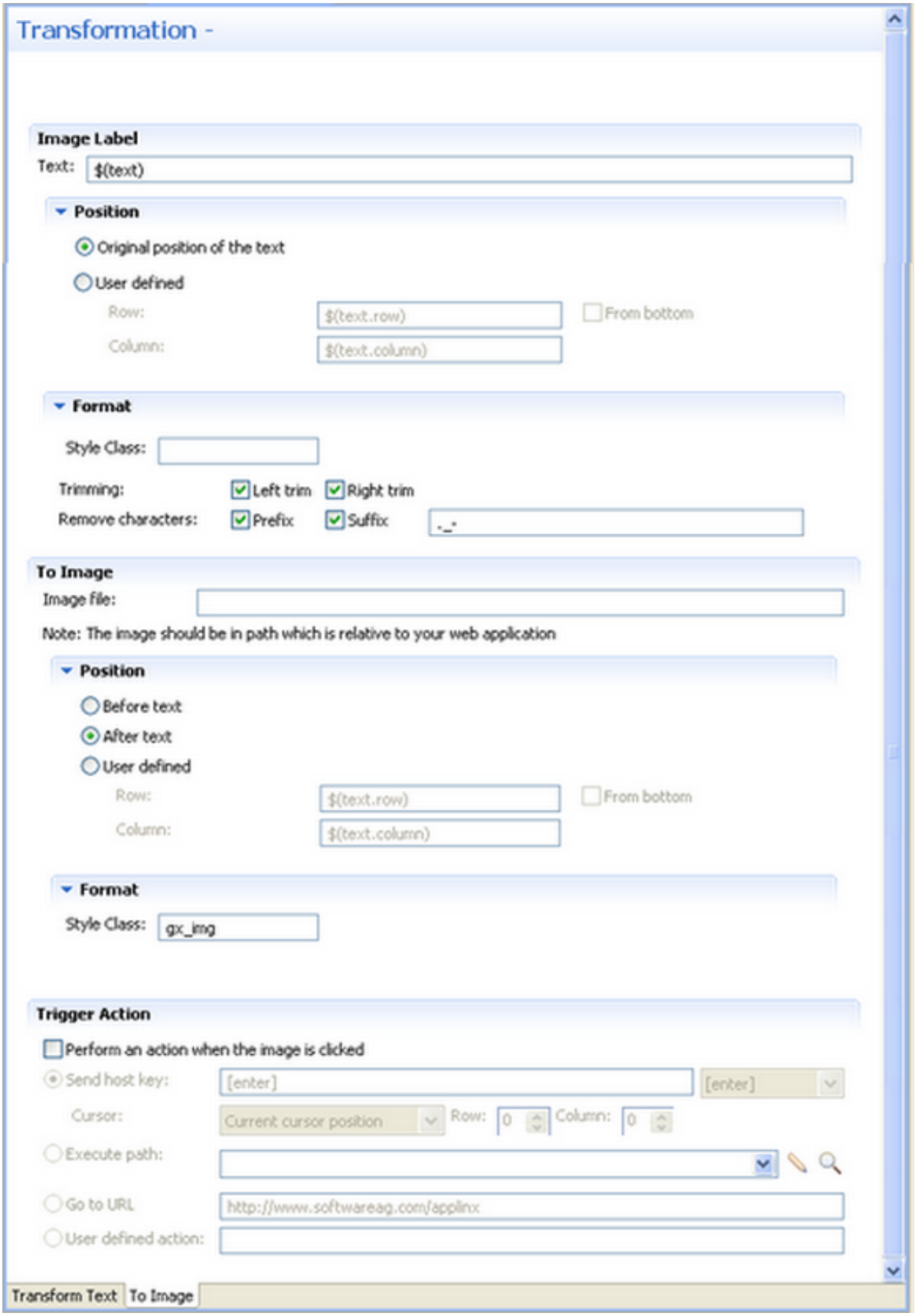
- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note:

When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

3 In the **To Image** Tab in the Editor:


Transformation -
Image Label
Text:
$(text)
Position
Original position of the text
User defined
Row:
$(text.row)
From bottom
Column:
$(text.column)
Format
Style Class:
Trimming:
Left trim
Right trim
Remove characters:
Prefix
Suffix
._-
To Image
Image file:
Note: The image should be in path which is relative to your web application
Position
Before text
After text
User defined
Row:
$(text.row)
From bottom
Column:
$(text.column)
Format
Style Class:
gx_img
Trigger Action
Perform an action when the image is clicked
Send host key:
[enter]
[enter]
Cursor:
Current cursor position
Row:
0
Column:
0
Execute path:
Go to URL
http://www.softwareag.com/applinx
User defined action:
Transform Text
To Image

1. Enter the text label for the image. By default the text is the original text and is represented by a variable - `$(text)`. Refer to Transformation Variables and their Attributes for further details about setting variables.
2. Define the position of the text. By default the position is the original position of the text. Select the User defined radio button to define the row and column (by default these values are the row and column of the original pattern: `$(text.row)` and `$(text.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
3. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.
4. Enter the image file name. It is possible to include references to variables, for example *$(text).gif*, and in this way use dynamic image names. Often, the image path is used in the Web application which is normally on a different machine and therefore it is recommended to write the relative path. Note that the default image, points to an image that exists in the framework's built-in Web applications.
5. Define the position of the image. By default the position is **Before text**. It is also possible to select **After text** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the original pattern: `$(text.row)` and `$(text.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
6. Define the format of the image. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the image element.
7. In the Trigger Action section define the action that is going to be performed once the image is clicked. It is possible to:
   - Send a host key - Select or enter the host key to be sent.
   - Execute a path - Browse to locate the folder where the path is saved. Select the path from the drop-down list.
   - Go to URL - Type in the relevant URL.
   - Enter a User defined action - Type in customized code to perform a customized action.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming a Text Pattern to a Button

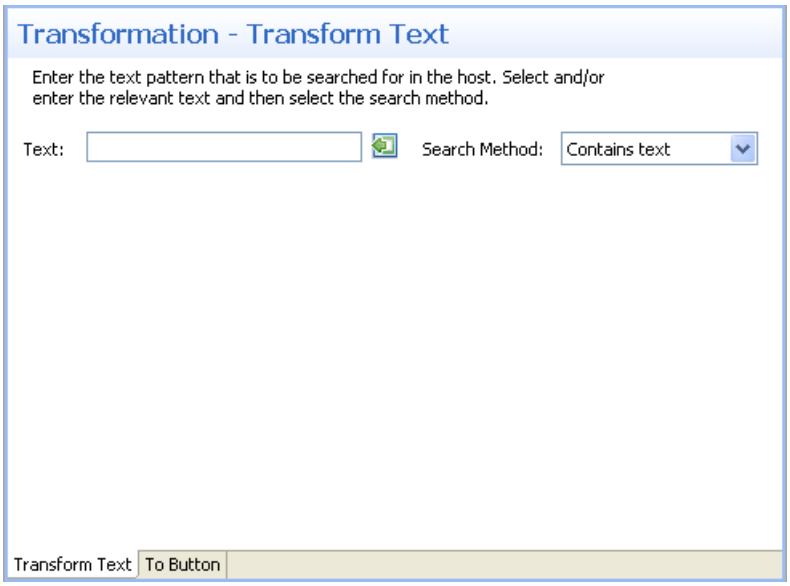
This transformation enables you to transfer a text pattern to a button.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a text pattern to a button

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Text** Tab in the Editor, enter the text pattern that is to be searched for in the host. Either type in the pattern or use the Capture from Screen feature to directly take a pattern from the host screen. Select the search method to use when searching for this text:

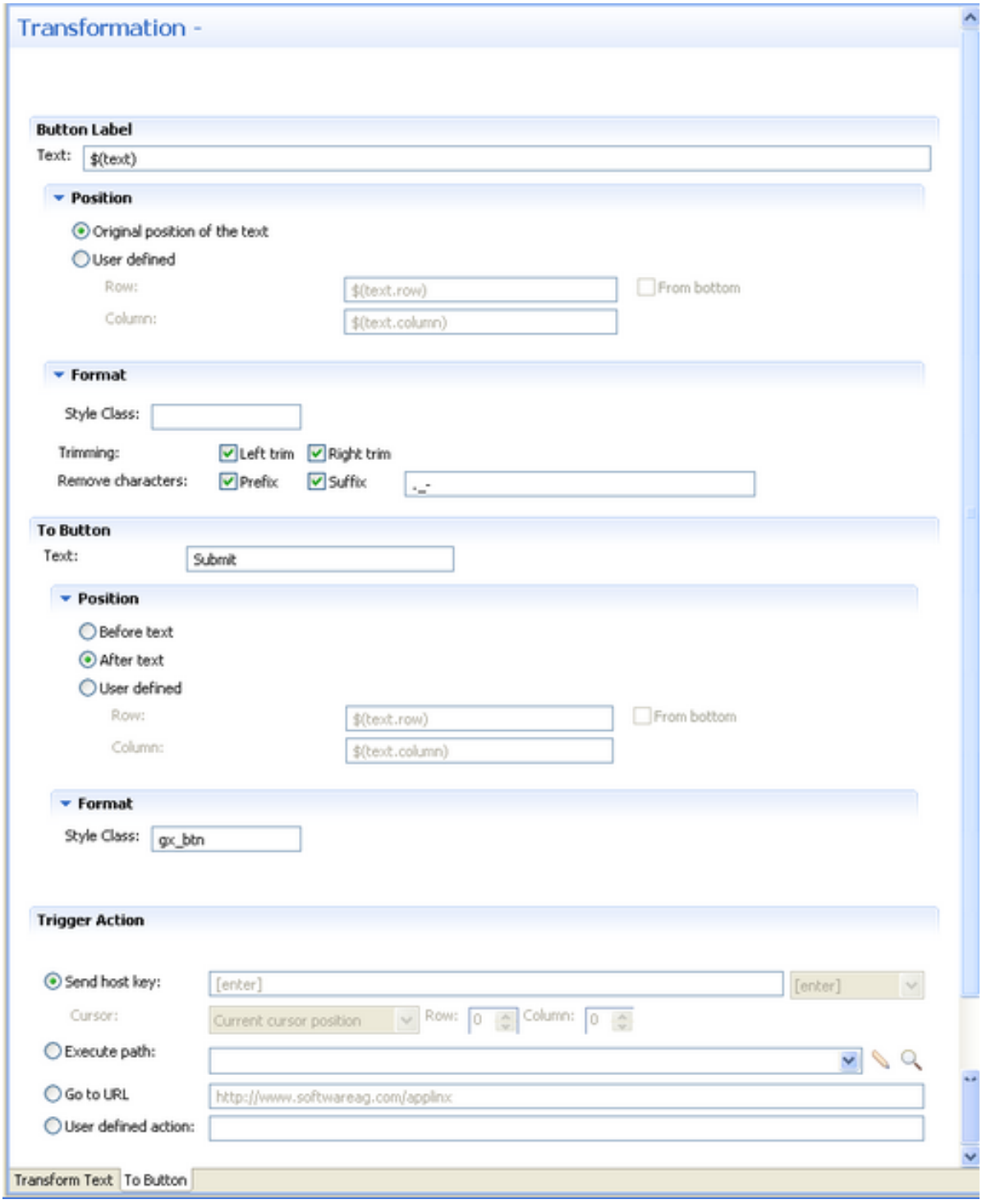
- Contains text: Searches for the given text anyway in the defined region (default).
- Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note:

When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

3 In the **To Button** Tab in the Editor:



1. Enter the text that will be displayed near to the button. By default the text is the original text and is represented by a variable - `$(text)`. Refer to Transformation Variables and their Attributes for further details about setting variables.
2. Define the position of the text. By default the position is the original position of the text. Select the User defined radio button to define the row and column (by default these values

are the row and column of the original pattern: `$(text.row)` and `$(text.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.

3. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.
4. Enter the text that will appear on the button.
5. Define the position of the button. By default the position is **Before text**. It is also possible to select **After text** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the original pattern: `$(text.row)` and `$(text.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
6. Define the format of the button. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text.
7. In the Trigger Action section define the action that is going to be performed once the image is clicked. It is possible to:
   - Send a host key - Select or enter the host key to be sent.
   - Execute a path - Browse to locate the folder where the path is saved. Select the path from the drop-down list.
   - Go to URL - Type in the relevant URL.
   - Enter a User defined action - Type in customized code to perform a customized action.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming an Input Field to a Text Field

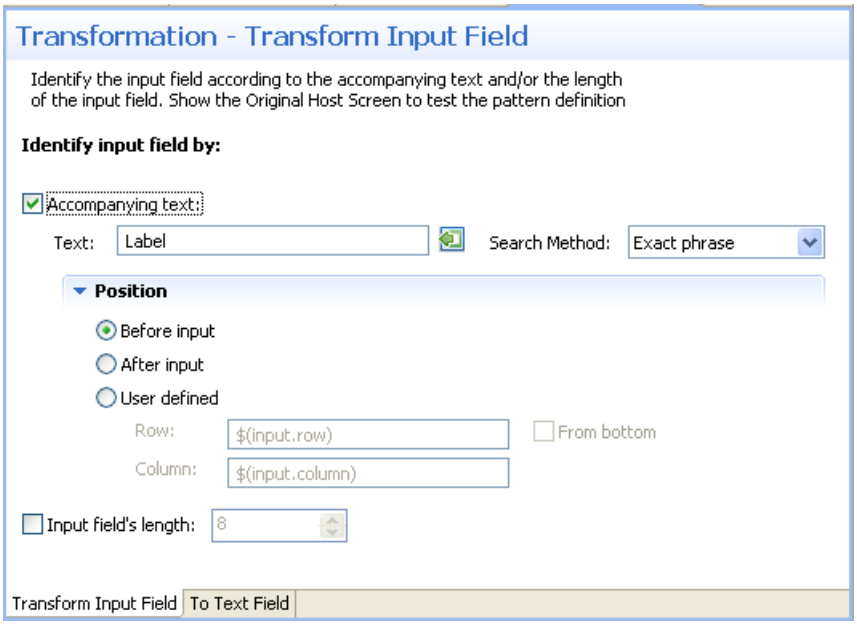
This transformation enables you to transfer an input field to a text field.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform an input field to a text field

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Input Field**Tab in the Editor identify the input field according to accompanying text and/or the length of the input field.

1. ■ Accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
   - Contains word: Searches for the given text as word(s) within the defined region.
   - Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

   Note:

   When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

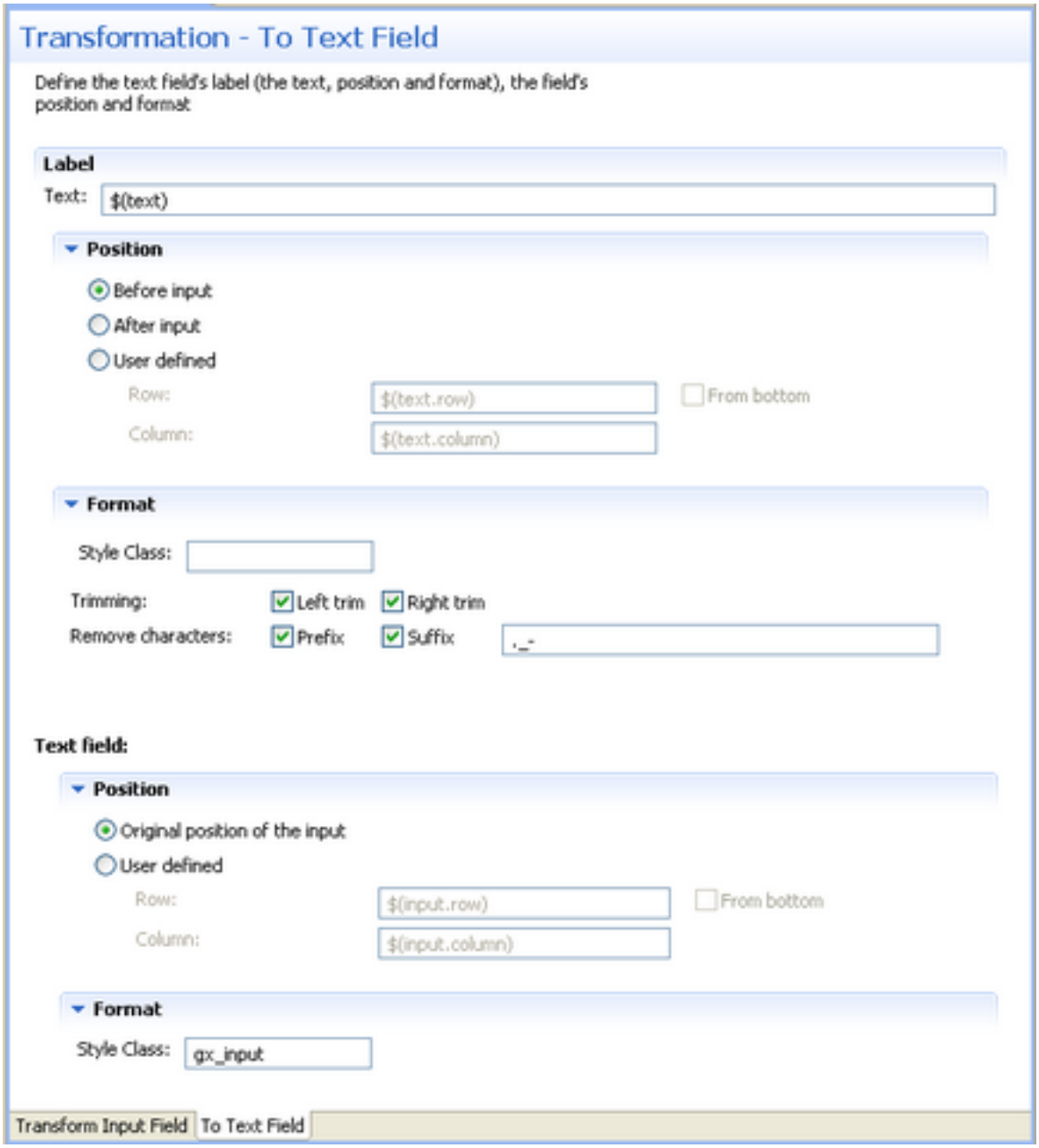
   When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

- Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.

2. Input field's length - select this check box to identify the input field according to the input field's length. The Length field will become enabled. Type in the length of the input field.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Text Field**Tab in the Editor define the label of the text field (text, position and format), as well as the position and the format of the text field:



1. Enter the label text. By default the text is the original text and is represented by a variable - `$(text)`.
2. Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.

3. Define the format of the label. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.
4. Define the position of the text field. By default the position is the original position of the input field. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the original pattern: `$(input.row)` and `$(input.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.
5. Define the format of the text field. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming an Input Field to a Combo Box

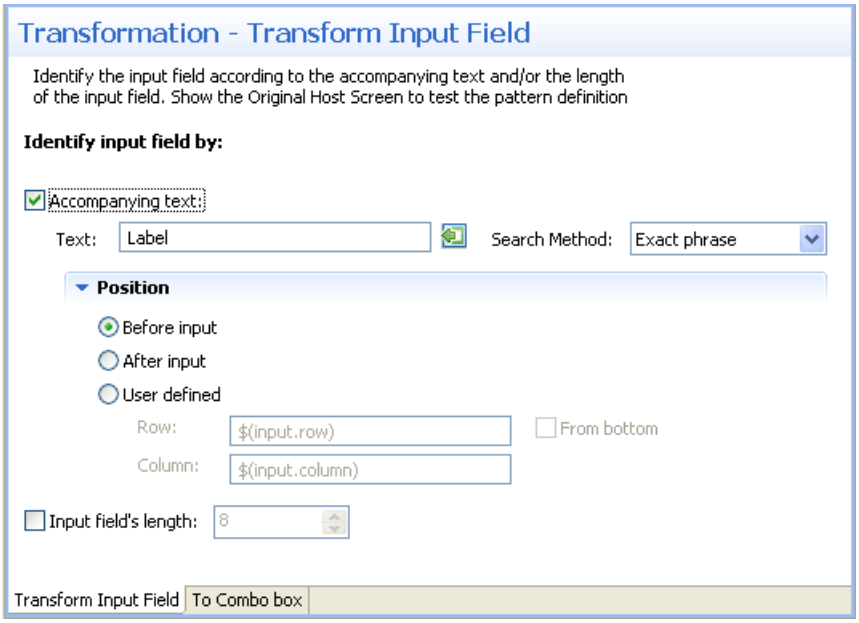
This transformation enables you to transfer an input field to a combo box.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform an input field to a combo box

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Input Field**Tab in the Editor identify the input field according to accompanying text and/or the length of the input field.

1. ■ Accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
   - Contains word: Searches for the given text as word(s) within the defined region.
   - Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

   Note:

   When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

   When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

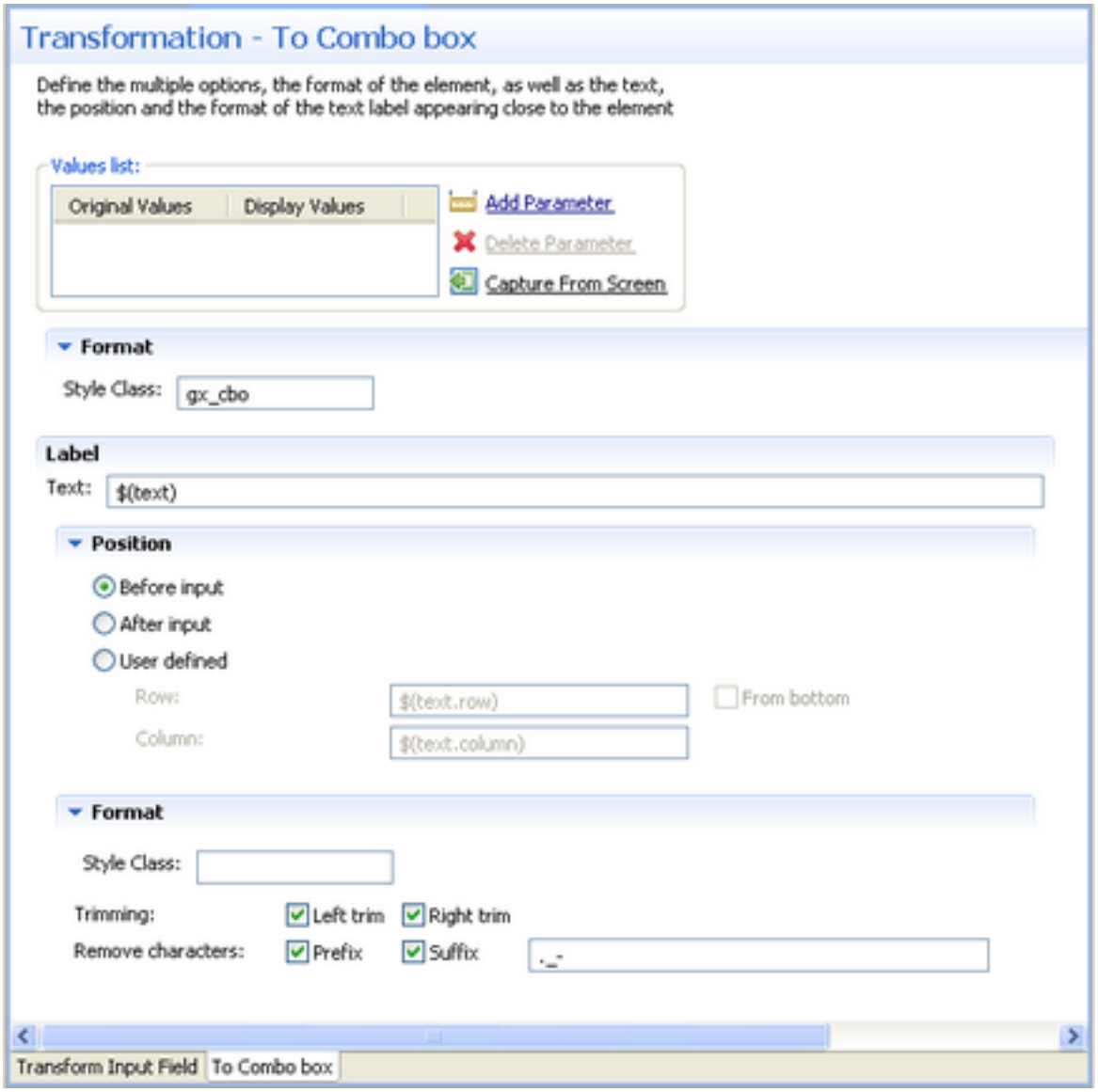
   ■ Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transform-

ation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.

2. Input field's length - select this check box to identify the input field according to the input field's length. The Length field will become enabled. Type in the length of the input field.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Combo Box**Tab in the Editor define the combo box values and label definitions:



1. Define the list of values for the combo box. Click the Add Parameter to add rows. Click on the cell in the Original Value column and either type in the original value or use the Capture from Screen feature to directly take the value from the host screen. Click on the cell in the

Display Values column and either type in the value to be displayed or use the Capture from Screen feature to directly take the value from the host screen. Use the Move Up and Move Down arrows to determine the order of appearance of the display values.

Define the format of the combo box: Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.

2. Enter the text that will appear before the combo box. By default the text is the original text and is represented by a variable - `$(text)`.
3. Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(text.row)` and `$(text.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
4. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming an Input Field to Radio Buttons

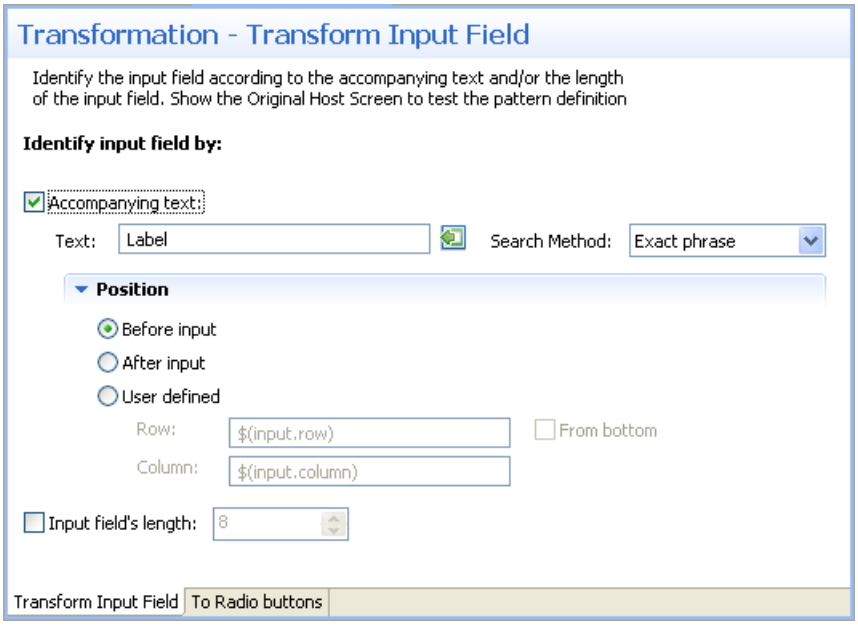
This transformation enables you to transfer an input field to a radio buttons.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform an input field to radio buttons

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Input Field**Tab in the Editor identify the input field according to accompanying text and/or the length of the input field.

1. ■ Accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
   - Contains word: Searches for the given text as word(s) within the defined region.
   - Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

   Note:

   When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

   When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

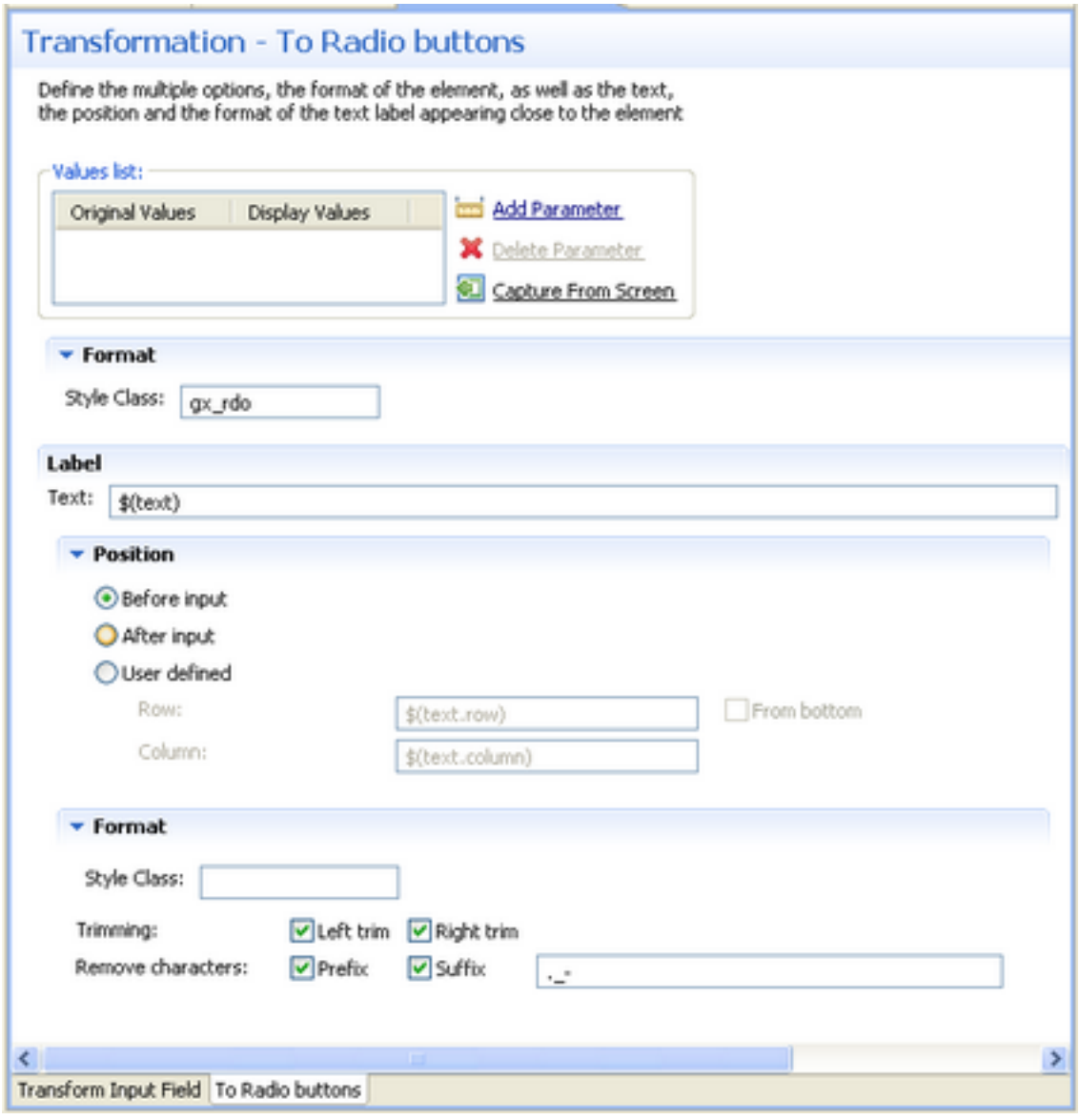
   ■ Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transform-

ation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.

2. Input field's length - select this check box to identify the input field according to the input field's length. The Length field will become enabled. Type in the length of the input field.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Radio Buttons**Tab in the Editor define the radio button values and label definitions:



1. Define the list of values for the radio buttons. Click the Add Parameter to add rows. Click on the cell in the Original Value column and either type in the original value or use the

Capture from Screen feature to directly take the value from the host screen. Click on the cell in the Display Values column and either type in the value to be displayed or use the Capture from Screen feature to directly take the value from the host screen. Use the Move Up and Move Down arrows to determine the order of appearance of the display values.

Define the format of the radio buttons: Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.

2. Enter the text that will appear before the radio buttons. By default the text is the original text and is represented by a variable - `$(text)`.
3. Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(text.row)` and `$(text.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
4. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming an Input Field to a Check Box

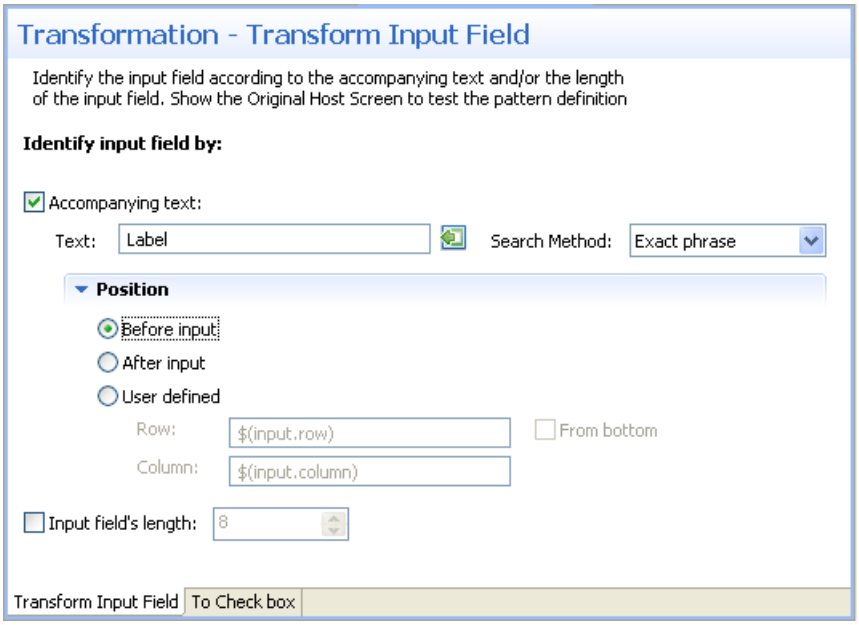
This transformation enables you to transfer an input field to a check box.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform an input field to a check box

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Input Field**Tab in the Editor identify the input field according to accompanying text and/or the length of the input field.

1. ■ Accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
   - Contains word: Searches for the given text as word(s) within the defined region.
   - Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

   Note:

   When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

   When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

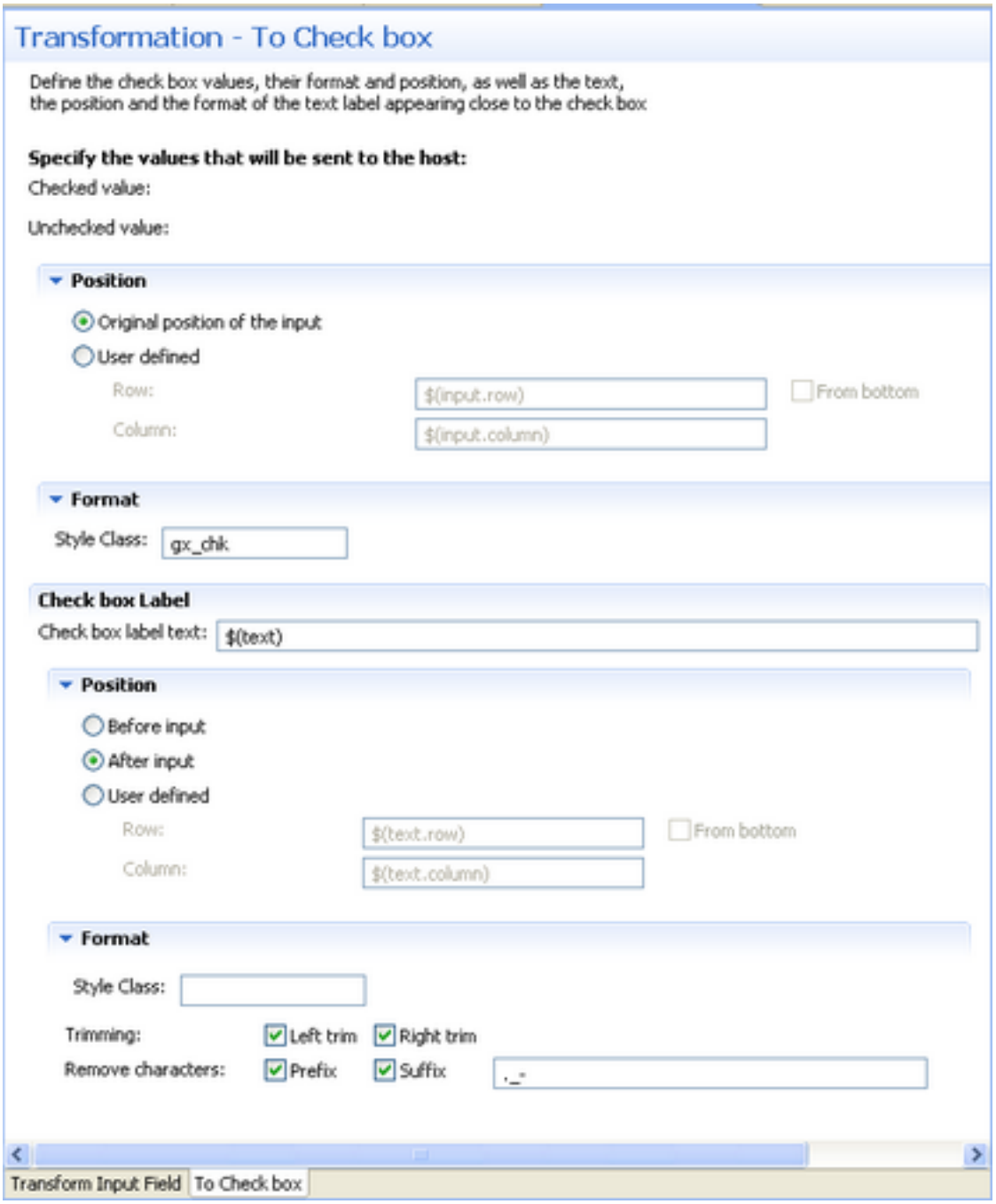
   ■ Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transform-

ation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.

2. Input field's length - select this check box to identify the input field according to the input field's length. The Length field will become enabled. Type in the length of the input field.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Check Box**Tab in the Editor define the check box values when selected and when not selected, the position and format of the check box as well as the text, the position and the format of the text label appearing close to the check box.



1. Define the values that will be sent to the host when the check box is selected, and when the check box is not selected.
2. Define the position of the field: By default the position is the original position of the input field. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input, to allow relative position calculations: `$(input.row)` and `$(input.column)`). Select **From bottom** to define the row and column number relative to the bottom of the screen.

3. Define the format of the check box. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.
4. Enter the text that will appear before the check box. By default the text is the original text and is represented by a variable - `$(text)`.
5. Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(text.row)` and `$(text.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
6. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming an Input Field with Values to a Combo Box

This transformation enables you to transfer an input field with a value to a check box. .

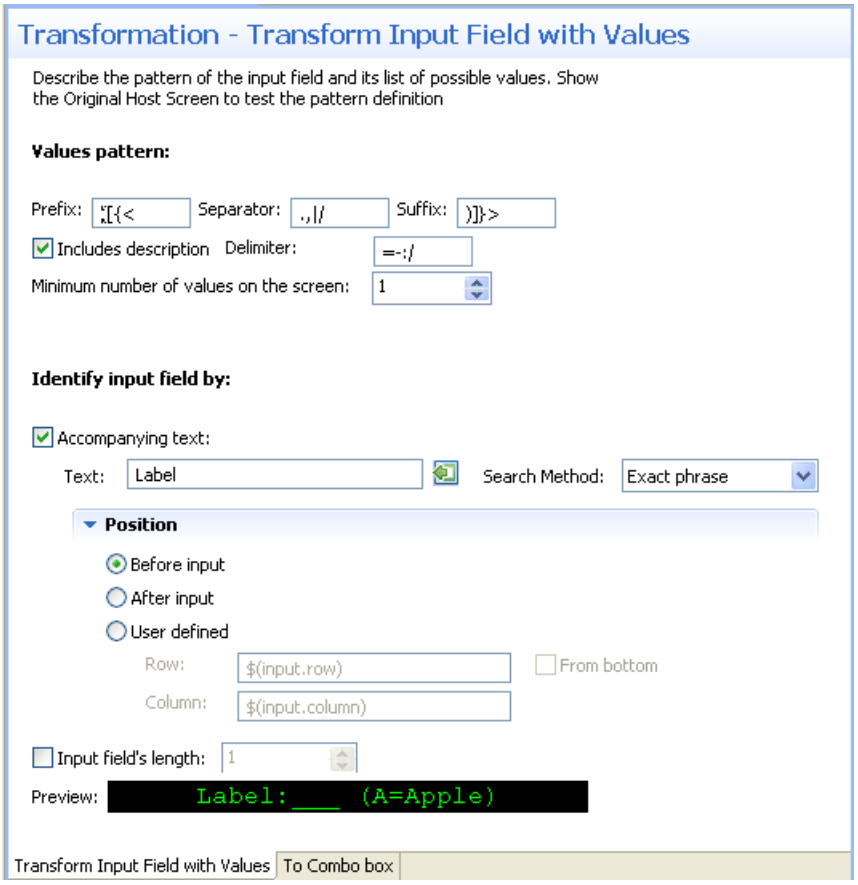
Refer to:

- Host Pattern Elements for an example of this pattern.

To transform an input field with values to a combo box

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .

2 In the **Transform Input Field with Values**Tab in the Editor define the pattern of the input field as well as whether to identify the input field according to accompanying text and/or the length of the input field:



1. Define the input field - the possible prefix, separator and suffix, indicate whether each value is preceded by text, the possible delimiter characters and the minimum number of values in the list of values.
2. Accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
   - Contains word: Searches for the given text as word(s) within the defined region.
   - Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

   Note:

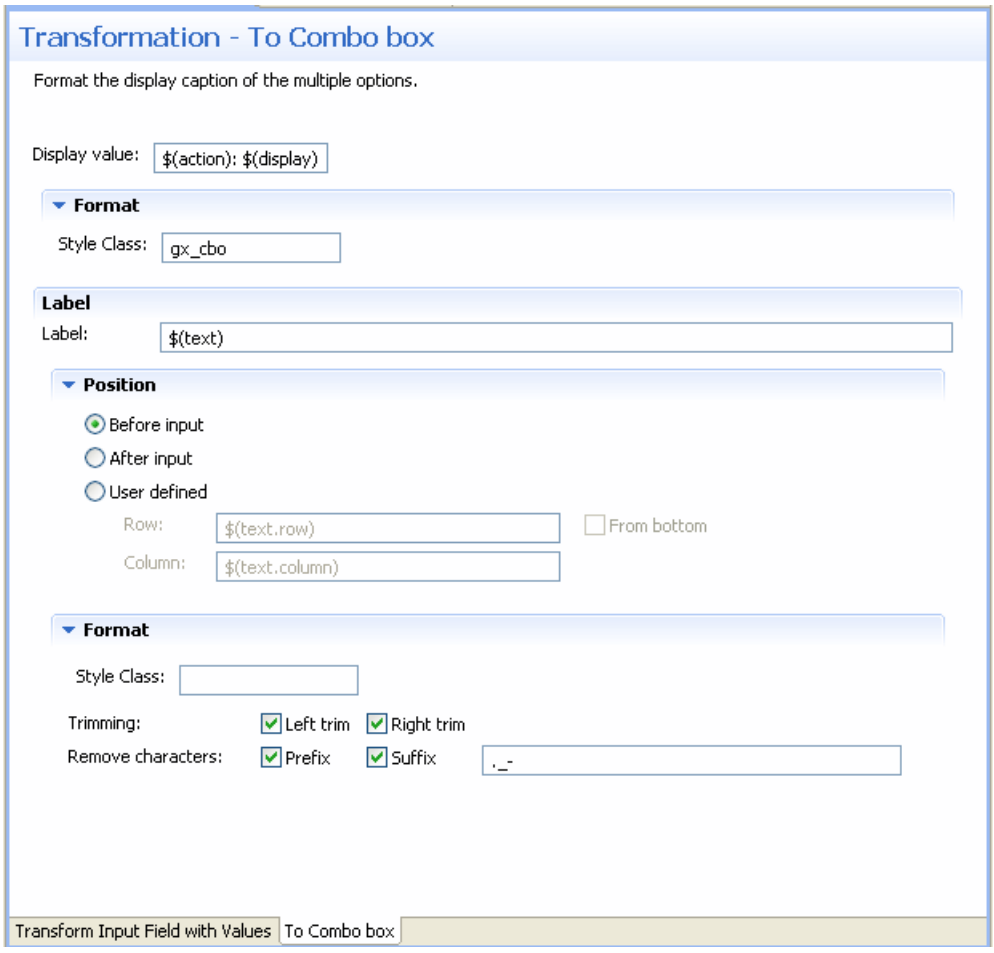
   When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

3. Define the position of the text. By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
4. Input field's length - select this check box to identify the input field according to the input field's length. The Length field will become enabled. Type in the length of the input field.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Combo Box**Tab in the Editor define the Combo box display value, the format of the combo box, as well as the label, the position and the format of the text label appearing close to the combo box:



1. Define the values that will be displayed. By default the display value is `$(action)`: `$(display)`.

   For example, if the pattern was (A=Apple, B=Banana, C=Coconut), the combo box value list will be:

   A: Apple

   B: Banana

   C: Coconut

   Changing the Display value field to `$(display) ($(action))` will result in the following value list:

   Apple (A)

   Banana (B)

   Coconut (C)

   Changing the Display value field to `$(display)` will result in the following value list:

   Apple

   Banana

   Coconut

   In all the above examples, the actual value sent to the host is the value represented by `$(action)`. For example, when Apple is selected, A will be sent to the host.
2. Define the format of the check box. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.
3. Enter the text that will appear before the combo box. By default the text is the original text and is represented by a variable - `$(text)`.
4. Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(text.row)` and `$(text.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
5. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents

of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming an Input Field with Values to Radio Buttons

This transformation enables you to transfer an input field with a value to radio buttons.

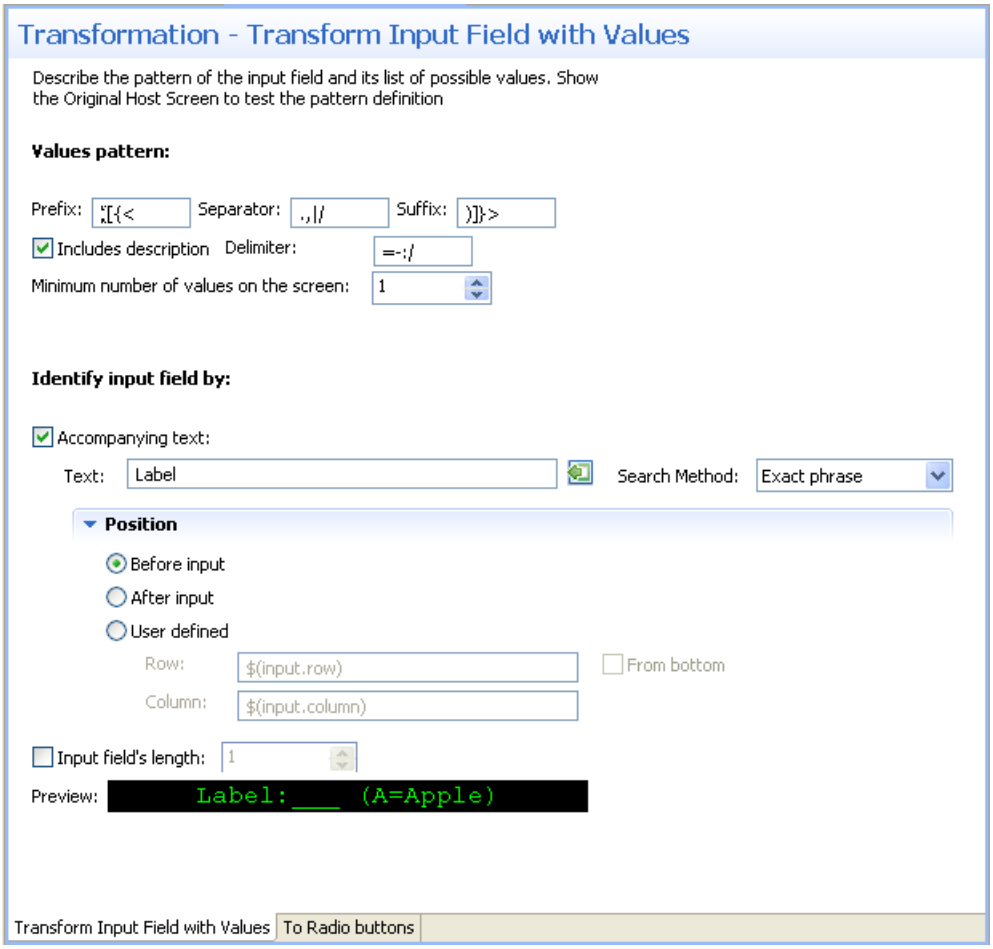
Refer to:

- Host Pattern Elements for an example of this pattern.

To transform an input field with values to radio buttons

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .

2 In the **Transform Input Field with Values**Tab in the Editor define the pattern of the input field as well as whether to identify the input field according to accompanying text and/or the length of the input field:



1. Define the input field - the possible prefix, separator and suffix, indicate whether each value is preceded by text, the possible delimiter characters and the minimum number of values in the list of values.
2. Accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.
   - Contains word: Searches for the given text as word(s) within the defined region.
   - Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

Note:

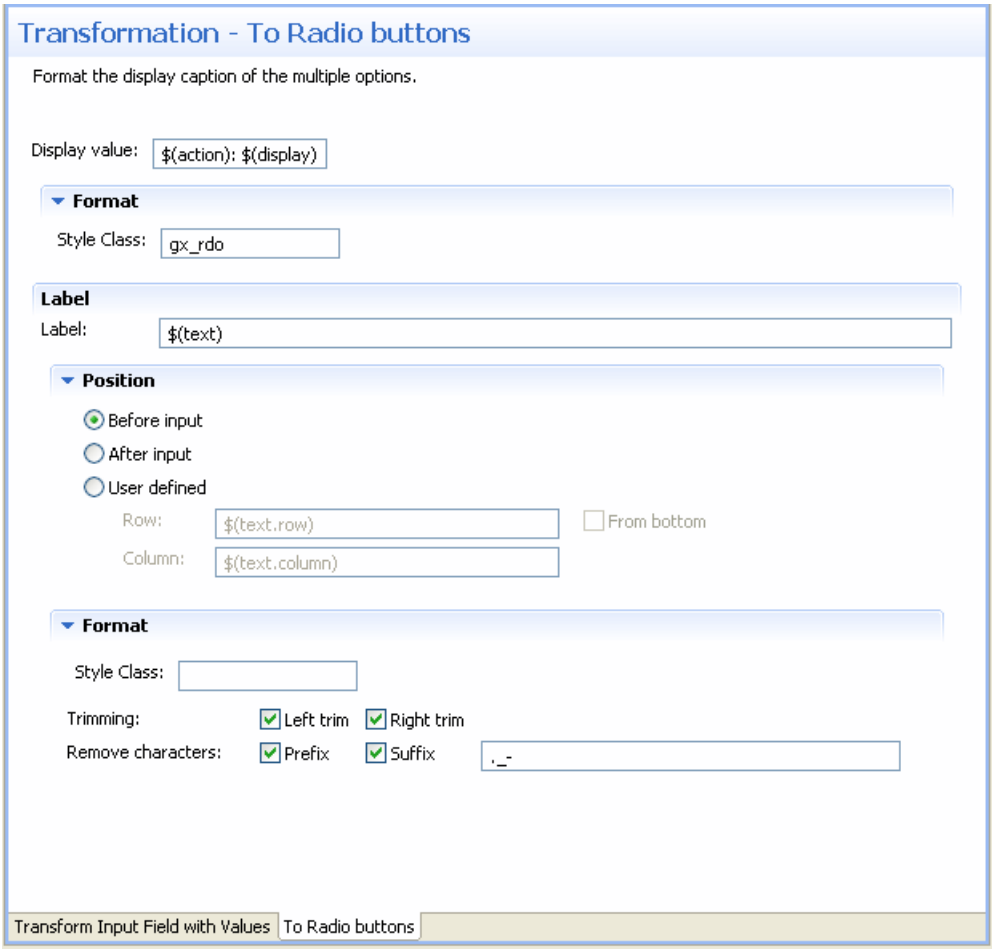
When selecting a different option, after defining a regular expression search definition, the value of the previous regular expression definition will be lost.

When a screen image is displayed in the Session view, the pattern definition will be marked in the screen image.

3. Define the position of the text. By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(input.row)` and `$(input.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
4. Input field's length - select this check box to identify the input field according to the input field's length. The Length field will become enabled. Type in the length of the input field.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Radio Buttons**Tab in the Editor define the radio buttons' display value, the format of the radio buttons, as well as the label, the position and the format of the text label appearing close to the radio buttons:



1. Define the values that will be displayed. By default the display value is `$(action): $(display)`.

   For example, if the pattern was (A=Apple, B=Banana, C=Coconut), the combo box value list will be:

   A: Apple

   B: Banana

   C: Coconut

   Changing the Display value field to `$(display) ($(action))` will result in the following value list:

   Apple (A)

   Banana (B)

   Coconut (C)

Changing the Display value field to `$(display)` will result in the following value list:

Apple

Banana

Coconut

In all the above examples, the actual value sent to the host is the value represented by `$(action)`. For example, when Apple is selected, A will be sent to the host.

2. Define the format of the check box. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.
3. Enter the text that will appear before the radio buttons. By default the text is the original text and is represented by a variable - `$(text)`.
4. Define the position of the text: By default the position is **Before input**. It is also possible to select **After input** or **User defined**. Select the **User defined** radio button to define the row and column (by default these values are the row and column of the input to allow relative position calculations: `$(text.row)` and `$(text.column)`). Refer to Transformation Variables and their Attributes for further details about setting variables. Select **From bottom** to define the row and column number relative to the bottom of the screen.
5. Define the format of the text. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the text element. Select the **Left trim** and/or **Right trim** check boxes to trim spaces from the right and left of the title. When necessary, enter the characters to be removed (when typing in more than one character, the contents of the field are related to as individual characters and not as a string of characters) and select **Prefix** or **Suffix** to determine the place of these characters.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Transforming a Date Input Field to a Calendar

This transformation enables you to transfer a date input field to a calendar component.

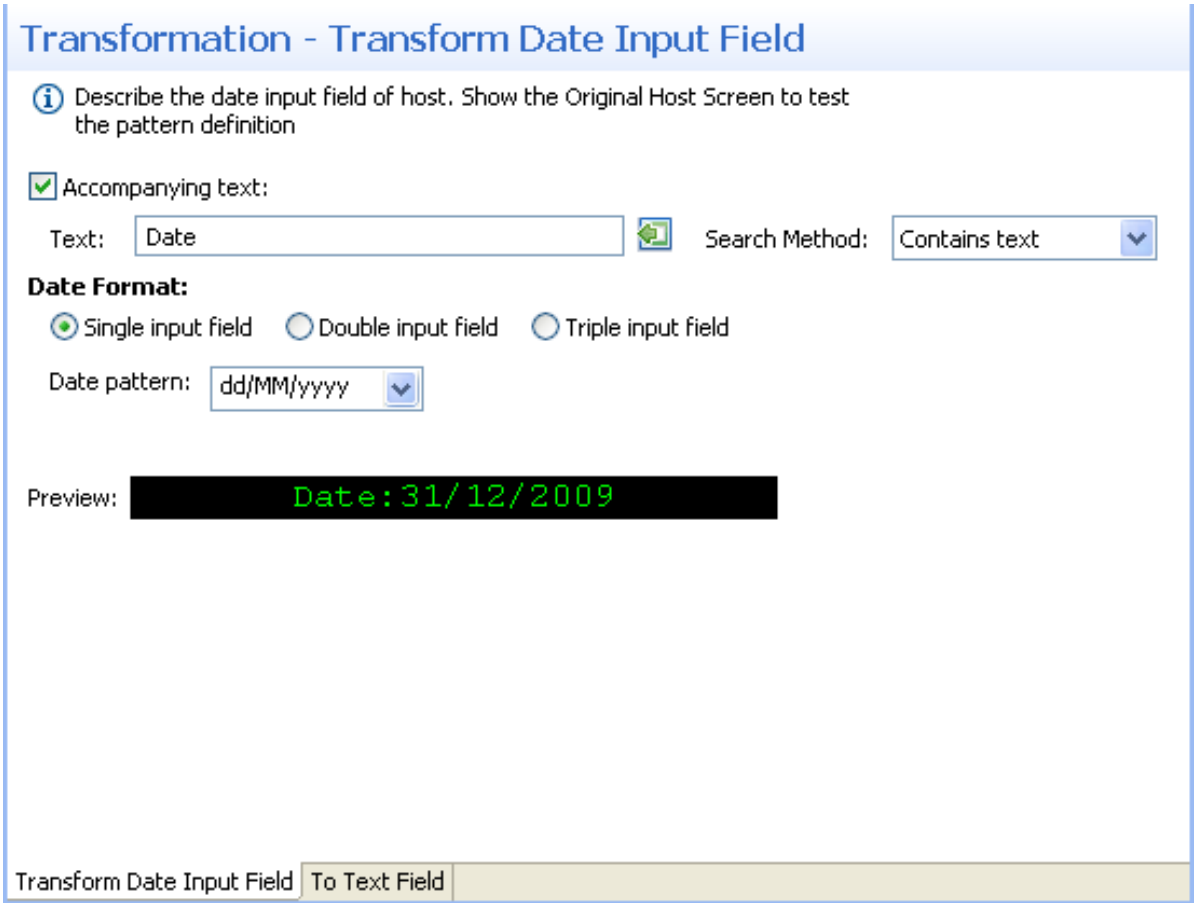
Note: The Calendar component language is determined according to the operating system. When ApplinX identifies that the Operating System is English, French, German, Spanish or Portuguese, the Calendar component is displayed in the identified language. Other operating systems are displayed in English.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a Date Input Field to a Calendar

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .



2 In the **Transform Date**Tab in the Editor determine whether to identify the date input field according to accompanying text and define the date format:

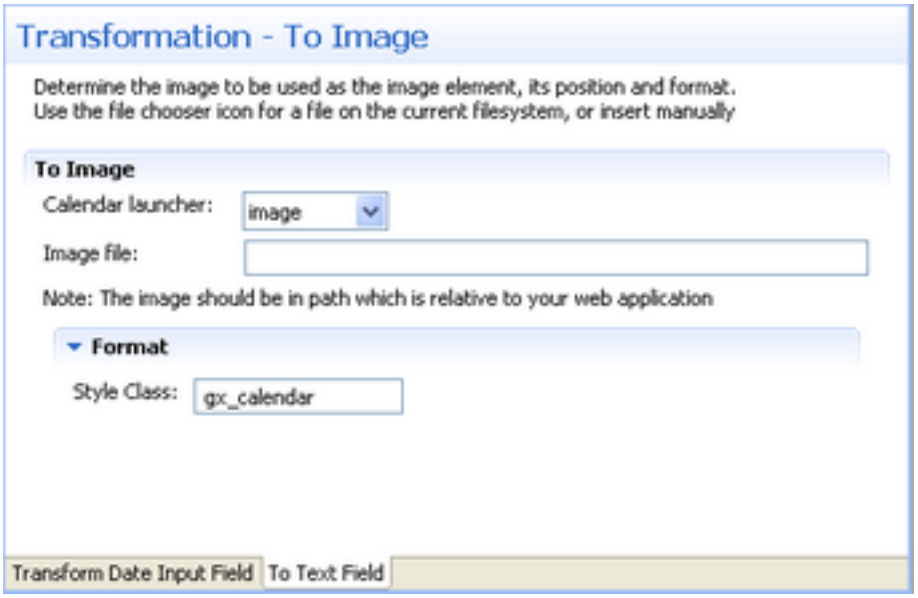
1. Identify the input field by the accompanying text - select this check box to identify the input field according to accompanying text. The text field will become enabled. Either type in the text or use the Capture from Screen feature to directly take text from the host screen. Select the search method to use when searching for this text:
   - Contains text: Searches for the given text anyway in the defined region (default).
   - Exact phrase: Searches for the given text within the defined region. The text must be preceded and followed by at least two white space characters.

- Contains word: Searches for the given text as word(s) within the defined region.
- Regular expression: Searches for the given pattern within the defined region. Refer to Regular Expression Syntax.

2. Define the date format - define whether the date in the host is in a single input field, in two input fields or in three input fields. Select the date pattern from the drop-down list of available patterns or type a different date pattern. Refer to Date and Time Patterns for further information regarding date formats.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Calendar**Tab in the Editor determine whether the calendar will be represented by an image, a button or a hyperlink. After determining the representation, define the relevant image file/text and format that will appear:



1. Click on the arrow of the Calendar Launcher drop-down list and select Button, Link or Image. When selecting Button or Link, enter the text that will appear on the button/link. When selecting image, select the image file or type in its path (it is possible to include references to variables, for example `$(text).gif`, and in this way use dynamic image names). Often, the image path is used in the Web application which is normally on a different machine and therefore it is recommended to write the relative path. Note that the default image, points to an image that exists in the framework's built-in Web applications.
2. Define the format of the element. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen. Note that the calendar component is not displayed in the HTML preview.

Transforming a Menu to Hyperlinks

This transformation enables you to transfer a menu to hyperlinks.

Refer to:

- Host Pattern Elements for an example of this pattern.

To transform a menu to hyperlinks

1 Create a new transformation as detailed in **Creating a Transformation** and select the relevant type of pattern .

2 In the **Transform Menu**Tab in the Editor define the identifying parameters of the menu:

1. Type in the string that appears before the leading token. The leading token is one or more characters that appear in front of the menu item description. These characters are often used by the host, to identify the menu item value. The leading token may be for example a number such as 1, 2, 3, or a letter such as a, b, c. Defining the string that appears in front of the leading token, helps identifying the menu pattern. By default the value of this field is two spaces. Refer to Host Pattern Elements for an example of this pattern.
2. Define the maximum number of characters that the leading token may be.
3. Define one or more possible delimiters. A specific menu will be identified only if it has exactly one type of delimiter, but the same transformation definition can match several different types of menus.
4. Type in the string that appears after the menu item. By default the value of this field is two spaces.
5. Select the minimum number of items required in order to identify the menu pattern.
6. Define where to search for the field where the menu item is selected: in the current cursor position, the first unprotected field, the last unprotected field, the previous unprotected field, the next protected field or in a user defined position.

Note: You can view and check the pattern definition by looking at the screen image displayed.

3 In the **To Hyperlinks**Tab in the Editor define the text that is going to be displayed: define the values that are going to be displayed, the format of the hyperlinks, and determine whether the field where the menu item is selected will be shown or hidden.

1. Define the values that will be displayed. By default the display value is `$(action)`: `$(display)`. Refer to Transformation Variables and their Attributes for further details about setting variables.

   For example, if the pattern was:

   1. Inventory Management

   2. Inventory Purchasing

   3. Sales Order Processing

   The display will be:

   1: Inventory Management

   2: Inventory Purchasing

   3: Sales Order Purchasing

2. Define the format of the element. Enter a name of a style class (pre-defined in the Framework) that will be used to determine the format of the element.
3. Select whether to hide the field where the menu item is selected or to show it.

4 Save the transformation.

5 **Map the transformation to a screen or screen group**.

6 Test the transformation: Select a screen/screen group and click on the **Entity** menu. Select **Open HTML Preview**. When there is an existing Web application, test the transformation by refreshing the relevant screen.

Mapping a Transformation to a Screen/Screen Group

Mapping transformations to screens or screen groups enables applying the same transformation to multiple screens or screen groups, and when necessary to different screen regions. Transformations are mapped to screens/screen groups in the Screen/Screen Group dialog box, Transformations tab. When deciding to use a transformation it is necessary to define a region in which to search for the original host pattern. The region can be an entire screen, a rectangle in a screen, a number of specific consequent rows or specific application fields.

To map a transformation to a screen/screen group

1 Open the relevant screen or screen group and display the Transformations tab.

2 Either click on a transformation that is in the list of transformations, or click Add Transformation, to add an additional transformation to this screen/screen group and select a transformation from the drop down list.

3 Select a Region type:

- Select **Anywhere on screen** to search the entire screen for the host pattern.
- Select **Row range** to search within specific consequent rows. Define the first and last row or use the Capture from Screen feature to select these rows. Select **From bottom** to define that the row numbers in the **Start Row** and **End Row** fields are calculated from the bottom of the screen.
- Select **Rectangle** to search for the host pattern within a defined rectangle. Define the beginning of the rectangle by its **Row** and **Column** and then define the end of the rectangle by its **Row** and **Column** (it is also possible to use the capture from Screen feature to identify a range (rectangle)).
- Select **Application field** to search for a specific application field and then select the field. It is possible to select a number of fields.

Overriding an Inherited Transformation

Transformations can be mapped to screens or to screen groups. Screens associated with a particular screen group will inherit from the screen group the field mappings and transformations mapped to that screen group. Sometimes, not all the transformations defined in the screen group are relevant to a particular screen and then it is necessary to override the inherited transformation and not use it. Another situation where you may need to override an inherited transformation is when you would like to apply the transformation to a different screen region than the screen region defined in the inherited transformation. In such a case, override the transformation (as detailed below), and then remap the transformation by adding it to the screen with the suitably defined screen region.

To override an inherited transformation

1 Access the relevant screen by either double-clicking on the screen in the repository or navigating to the relevant screen in the session view and clicking Open screen in the right-click pop-up menu.

2 Select the **Transformations** tab.

3 Focus on the relevant transformation.

4 Click on the **Disable Transformation** hyperlink.

Note: Redefine that this transformation is to be used in the screen by clicking on the **Enable Inherited Transformation** hyperlink.

16 Procedures

What are Procedures and Procedure Groups?

An ApplinX Procedure is a well-defined encapsulation of a complete process, and contains process input arguments, process output arguments and the process definition itself. A procedure group is a container of several procedures. In ApplinX, Procedure Groups are equivalent to Web services, and Web methods that implement these Web services are called Procedures. Using Procedures, any enterprise application can retrieve information from a host application and input data to a host application. There are a number of procedure types in ApplinX:

- Path Procedure: encapsulates a process of navigation in host screens, collecting data or submitting data.
- Flow Procedure: encapsulates a complex process that can combine host sessions and other data sources: databases, host transactions (RPC), other web services.
- Web Procedure: encapsulates a process of navigating and selecting elements in the Web.
- Program Procedure: encapsulates a host transaction (in COBOL or RPG), invoked via RPC and not via the screens layer.
- External Web Services: encapsulates a non-ApplinX web service, invoked via SOAP.

Procedures can be exposed for external use using Web services, procedure clients and ApplinX Base Object.

- How to know which Type of Procedure to use?
- Path Procedures
- Flow Procedures
- Web Procedures
- Program Procedures
- External Web Services
- Procedure Groups
- Procedure Clients
- Deploying a Procedure Group to Integration Server
- Debugging Procedures

How to know which Type of Procedure to use?

The Path procedure is similar to the Flow Procedure. To know which one to use, ask yourself the following:

- Do you need to encapsulate a transaction from a single host session (this is the most common case)? Use a Path procedure.
- Do you want to execute the encapsulated transaction from the ApplinX Base Object API (ABO)? Use a Path procedure.
- Do you need to integrate data from different sources, such as database, several host sessions, external web services or RPC-based sessions? Use a Flow Procedure.
- Do you need to integrate data from the Web? Use Web Procedures.

Path Procedures

Path procedures on the one hand allow navigating between host screens and can be used for navigation, data gathering or transaction execution on a single host session. On the other hand, it is a Procedure, and can therefore be activated as a Web service operation, using a generated procedure client or called by another flow procedure. Procedures, provide a rich variety of logical nodes and expressions that make the host navigation very flexible without needing client side code.

- Creating a Path Procedure
- Path Procedure Methodologies
- The Path Procedure Editor
- Defining Procedure Inputs and Outputs
- Procedure Input and Output Attribute Types
- Working with Procedure Nodes
- General Nodes (Relevant for Flow, Path and Web Procedures)
- Path Procedure Nodes
- Using the Mapper to Map Source Elements to Target Elements
- General Expressions (relevant for Flow, Path and Web Procedures)
- Path Procedure Expressions
- Running and Debugging Procedures
- Path Procedure Failure Log

Creating a Path Procedure

A Path procedure can be created either via the Session or the Repository. When creating the Path procedure in the Session, you use the Path toolbar to record the path and then you can edit the path in the Path procedure dialog box. When creating the Path procedure via the Repository, you need to manually define the Path procedure using the available nodes. It is recommended to create the Path procedure via the session and then fine-tune and handle errors.

To create a Path Procedure via the Session View

1 Navigate to the first screen from which you want the recording to commence.

2 Click Record on the Toolbar.

3 Navigate to the various screens to record the relevant path.

4 Click on the Define Inputs icon to mark all the unprotected input fields in purple. Click on one of the marked fields to define it as an input. The field will be marked orange.

5 Click on Define Procedure Outputs to mark all the protected and unprotected fields. Click on a field to define it as an output in the Path procedure. The field will be marked blue.

6 If you want to record a step that repeats again and again until a certain condition is reached, click the loop icon, and follow the instructions in the wizard. This kind of definition can be applicable for collecting data of a host table.

7 To end a recording click Stop . The End Path Recording dialog box is displayed.

8 Enter a name for the path, and click Finish. The Editor will open where you can manually edit and add nodes.

Refer to the **Procedures** section for more information about Procedures.

To create a Path Procedure from Repository

1 Create a new Path Procedure via File>New>Other...>Software AG >Entities>Procedure.

2 The New Path procedure wizard is displayed. Enter a name for the Path procedure.

3 Define inputs and outputs in the bottom panel of the screen.

4 Define nodes. It is recommended to always create a Try/Catch node in order to catch exceptions within the procedure.

> **Note:** An image of the Procedure tree can be saved as a PNG file for later reference, by right-clicking on the Procedure root node, and selecting **Save as Image...**.

5 Save the procedure.

6 Right-click on the procedure in the ApplinX Explorer and select Run to run and test the procedure, or select Debug to open the Debug window and debug the procedure.

To Convert Old Paths or Path Wrappers to Path Procedures

- Old Paths are deprecated and are completely replaced with Path Procedures. Old Paths can still be used in runtime but cannot be edited or updated. Paths created in previous ApplinX versions will require conversion to the new Path Procedures using the conversion tool (right-click on a Path and select Convert).Refer to Migration for further details.

Path Procedure Methodologies

This section provides tips and methodologies that may be helpful when incorporating Path Procedures in your application.

Retrieving Field Attributes and Host Screen Properties

When executing Path Procedures, you may require retrieving field attributes or screen properties. One example for this may be collecting all positive values from a numeric list where all positive numbers are colored green and negative numbers are colored red. Another example is when submitting certain values to the host a modal window may appear which would require us to perform a slightly different navigation than usual.

The required field attributes and screen properties can be retrieved using the "Field Attributes" and "Screen Properties" expressions.

To retrieve field attributes and screen properties:

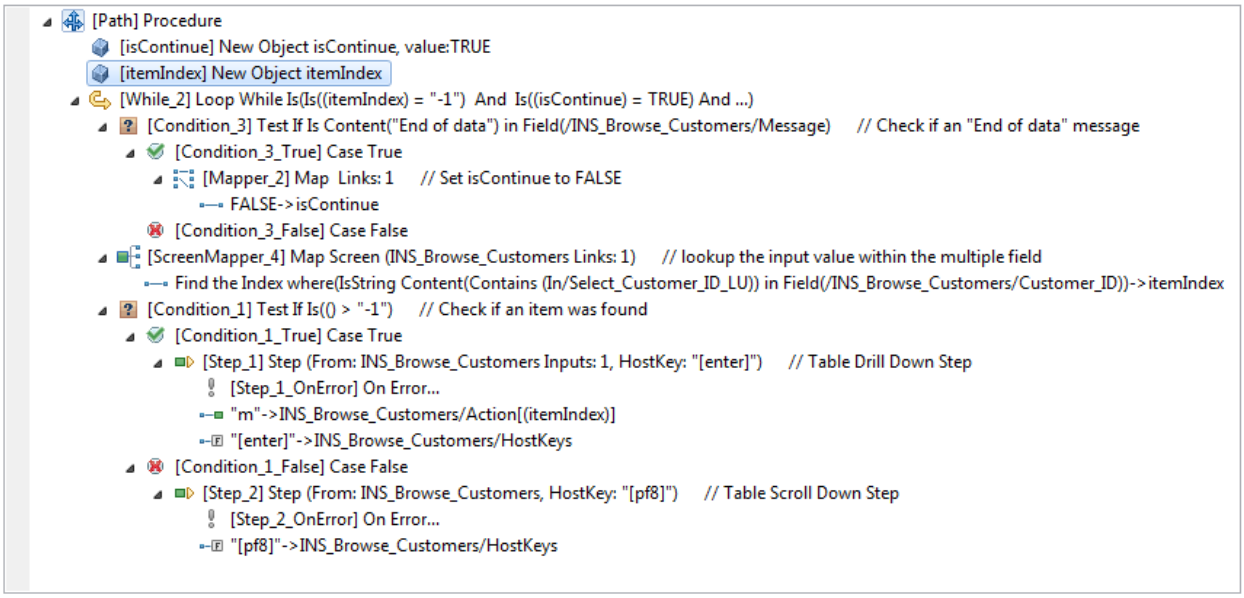
1. Right-click on the Expressions node in the mapper area, or if you are creating a condition, click on the Expr link and select Screen Data>Field Attributes/Screen Properties.
2. When working in a Mapper area, map the attribute or property to the appropriate variable.

   From within a condition, select the relevant attribute or property.

Searching for a Specific Value in a Multiple Field

When you need to search for a specific value within a host table that spans over several screens, what you actually need to do is to search for a specific value within a multiple field. A possible scenario may be, for example: A web page that contains a table with data collected from several screens. When clicking on a specific line, you would like to drill down and see additional details. The problem is having collected several screens on one web page, the selected line doesn't appear on the current host screen anymore - you need to look for it. In your ApplinX session, you need to scroll through the table again, and look for the particular data or a unique ID from the selected line on the web page. In order to do this, your Path procedure requires two loops: One loop, to scroll through the table and another loop, to search each line for the selected value.

This can be implemented using the Find Field Index function. This function allows you to check a single condition within a multiple field, testing each field and returning the index of the first field that matches the condition.



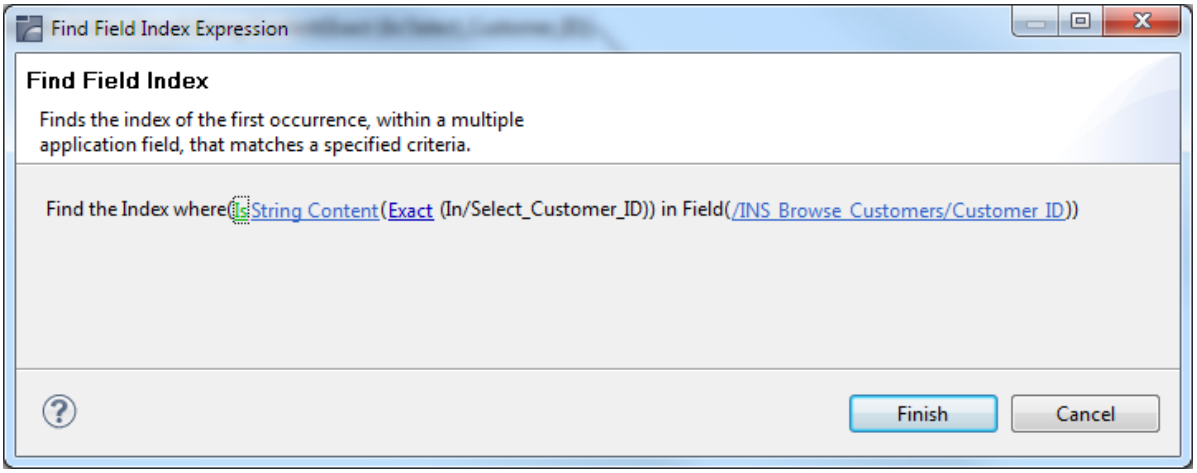
The loop will continue iterating through the table as long as both of these conditions are met:

- The `itemIndex` parameter is set to -1. This means no match to the search value has been found yet.
- The `isContinue` parameter is set to True. This parameter will be set to false when the end of the table is reached. This will prevent creating an infinite loop when no match is found.

The loops are defined as follows (see screen capture above):

1 The [Condition_3] condition checks whether you have reached the end of the table. When the end of the table is reached, IsContinue is set to False.

2 Configure the Find Field Index parameters:



- Attribute type: Select an attribute type, in this case String Content. Other possible attributes include: protected, hidden, intensified, reversed video, background color, foreground color, blinking or underlined.
- Select comparison type: You can check whether a string value is part of the field content or you can look for an exact match. Numeric values can be compared (less than, equals, greater than, etc) to the field's numeric value.
- Select the screen and multiple field whose content you wish to check.

Using Find Field Index, [ScreenMapper_4] searches for the value inside an entire table column (multiple field) and maps the result (the matched index) to the itemIndex parameter previously created. When no result is found, the mapping is not performed.

3 The [Condition_1] condition checks whether a match was found by checking if itemIndex is bigger than -1 (which means a result has been found). If a match was found, use itemIndex to implement a drill down action. If a match was not found, perform a scroll down action to continue the search.

Path Procedure Failure Log

ApplinX enables generating a log that includes debug data regarding procedures that fail in runtime. The log includes a snapshot in ASCII characters of the screen that caused the failure.

To log data regarding procedures that fail in runtime

1 In the <ApplinX home>\applinx-web\WEB-INF\config\log\gxlog_config.xml file under the procedure runtime error category tag, change the property "`proc.rt.err.log.level`" from "`off`" to "`debug`".

```
<!--  procedure runtime error -->
    <Property name="proc.rt.err.log.level">off</Property>
    <Property name="proc.rt.err.charset">UTF-8</Property>
    <Property name="proc.rt.max.file.size">1MB</Property>
```

2 Restart the ApplinX Server.

The log will be created in the <ApplinX home>/host-applications/<APPLICATION_NAME>/log directory, and the file names will have the following format: debugging_error_in_%I_%t.log, %I being identifying information about the user and procedure name, and %t being the timestamp. The location and name of the file can be changed in the gxlog_config.xml file.

Flow Procedures

Flow Procedures are available only for SOA licensed ApplinX servers. They are able to execute other types of ApplinX procedures in addition to being able to connect to databases and perform operations such as SELECT, Execute, Commit and Rollback. This allows retrieving data from sources other than host-screens.

- Creating a Flow Procedure
- The Flow Procedure Editor
- Defining Procedure Inputs and Outputs
- Procedure Input and Output Attribute Types
- Working with Procedure Nodes
- General Nodes (Relevant for Flow, Path and Web Procedures)
- Flow Procedure Nodes
- Using the Mapper to Map Source Elements to Target Elements
- General Expressions (relevant for Flow, Path and Web Procedures)
- Running and Debugging Procedures

Creating a Flow Procedure

When creating a new flow procedure, it is important to first define your goals and what tools you need in order to reach these goals. These tools may include paths or Data Structures (that represents inputs or outputs), programs or a database.

To create a Flow Procedure

1 Create a new Flow Procedure via File>New>Other...>Software AG >Entities>Procedure.

2 Define inputs and outputs in the bottom panel of the screen.

3 Define nodes. It is recommended to always create a Try/Catch node in order to catch exceptions within the procedure.

Note: An image of the Procedure tree can be saved as a PNG file for later reference, by right-clicking on the Procedure root node, and selecting **Save as Image...**.

4 Save the procedure.

5 Right-click on the procedure in the ApplinX Explorer and select Run to run and test the procedure, or select Debug to open the Debug window and debug the procedure.

Refer to the **Procedures** section for more information about Procedures.

Web Procedures

The Web Page Integration feature enables simulating web browser activity and exposing it as a standard web service or integrating it with other ApplinX procedures. Web browser activity is simulated within ApplinX by recording and entering/capturing relevant web content using Web browser based tools. The web content is used by the new type of Procedure - the Web Procedure. This Procedure is specifically designed to enable integrating the User Interface (UI) of Web Pages within ApplinX, taking advantage of the flexible and dynamic capabilities incorporated within ApplinX Procedure infrastructure. The Web procedure can be exposed as a service in the same way the existing ApplinX procedures such as Path and Flow Procedures are exposed.

The following topics will help you to create and work with Web Procedures:

- When your network requires defining a proxy, set the proxy Hostname, Port, Username and Password in the `WebProcedureConfig` section in the *<ApplinX installation>/config/gxconfig.xml* file. Refer to
- Create a new ApplinX application (including host and repository). Note that when creating a new application, you are required to enter a host. If you are only using the Web Page Integration solution, and are not integrating with other ApplinX components, this may seem to you unnecessary. Due to technical limitations, it is currently mandatory to define a host as part of the process of creating an application. As such, we recommend selecting a host from the list of predefined hosts.
- Create a new Web Procedure
- Running the Web Procedure
- Editing the Web Procedure
  - Adding a GoTo node, to Navigate to a URL
  - Adding conditional logic to the web procedure
  - Adding a New Element to an existing Web Page

- Handling Elements whose XPath has changed
- Working with dynamic XPaths
- Checking whether an Element Exists within a Web Page
- Retrieving an Element's Value

You may also want to read more information regarding the following topics:

- What is an XPath?
- Web Procedure Nodes including: GoTo URL Node, Web Page Node, Enter Text Node, Select Node and Click Node.
- General Procedure Nodes
- Web Procedure Expressions
- General Procedure Expressions

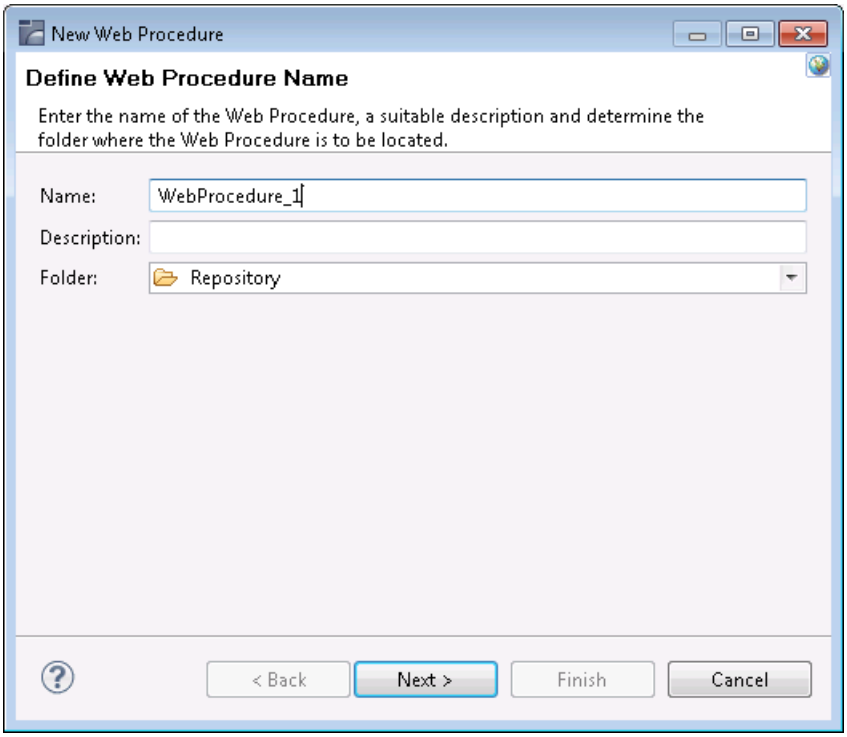
Creating a Web Procedure

As part of the process of creating the Web Procedure you will record and capture the elements to be used in the Web Procedure. Before creating a new Web Procedure, it is important to first decide and plan what the scenario is that you would like to record, and also determine which elements you would like to capture and later use in the Web Procedure.

To create a Web Procedure and capture the elements from within your browser session:

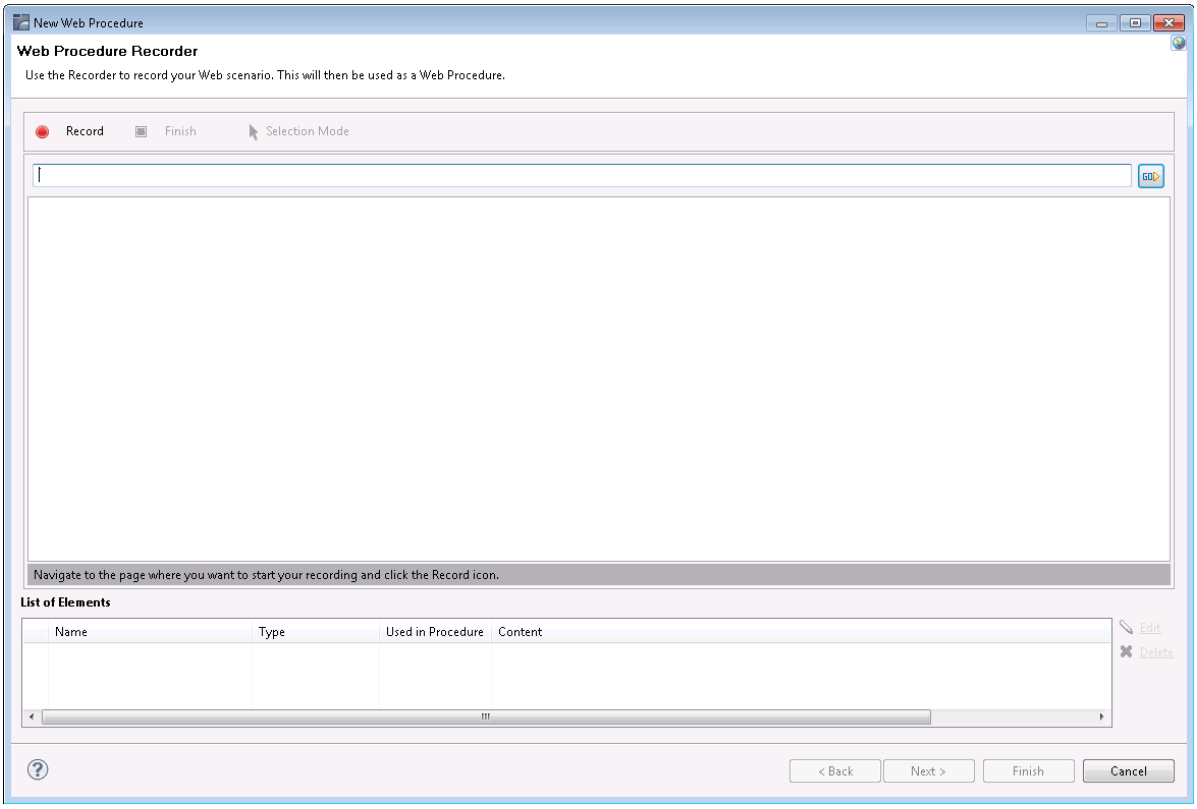
1 Select the relevant application or the Root node of the application.

2 In the **File** menu, select **New>Entity>Web Procedure…**. The *Web Procedure wizard* is displayed.



3 Enter a name for the Web Procedure, a suitable description and determine the folder where the procedure is to be located.

4 Click **Next**. The *Web Procedure Recorder* is displayed. The Recorder is comprised of a customized toolbar, an address bar, the browser area where the Web page content is displayed and a summary of the list of captured elements.



5 Enter the name of the URL and click the **GO** button or press ENTER to navigate to the URL where you would like to begin recording.

6 **Recording:**

Click on the **Record** button. When recording there are two possible modes: **Navigation mode** and **Selection mode**. Use the Navigate mode to navigate through the web pages as in any Web browser, and record inputs and actions such as clicking on buttons or links, entering data. Every action you do is recorded and can later be edited in the Web Procedure editor.Use the Selection mode to "freeze" the page, so you can capture outputs and inputs, single elements or a list of similar elements. When in the Selection mode, you can no longer navigate to other pages until you return to Navigation mode.

7 **Selecting and Capturing Elements:**

1. Entering the Selection Mode:

   When you want to select and capture elements on a specific page, click on the **Selection Mode** button 

.

2. Understanding the highlight colors:

   Light blue: When you move the mouse over different elements within the browser, the color of the element changes to light blue to indicate that this is the element which will be selected if you now click on this element.

Orange: Once you click on an element, it is highlighted in orange.

Dark blue: Captured elements are colored in dark blue.

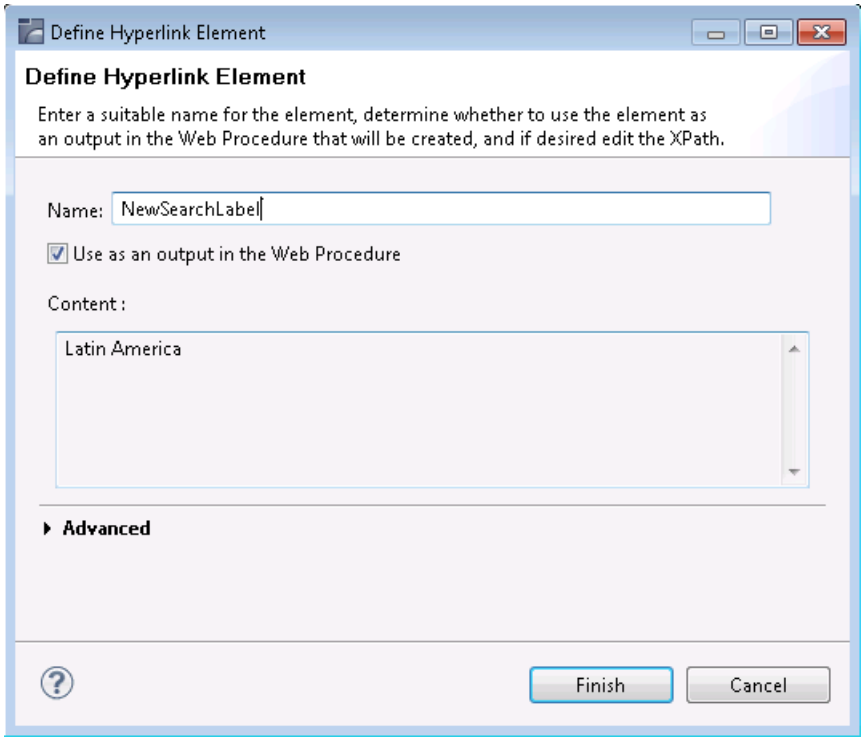
3. Selecting single elements and lists of elements:

   Within a page, you can capture either a single element or a list of elements. When entering the Selection mode, by default, the Recorder is set to capture single elements, by clicking once on an element. When you would like to capture a list of elements, click on the List button on the toolbar 

, then click on one element within your list and then on a second element. Before clicking on the second element, hovering over an element highlights in a light blue color the complete list of elements that will be captured once you click on the second element. Ensure that the Single button is pressed in when you wish to select single elements, and that the List button is pressed in when you wish to select a list of elements.

4. Capturing an element:

   Click on a required element. The *Define Element* dialog box is displayed and the element color within the browser changes to orange.



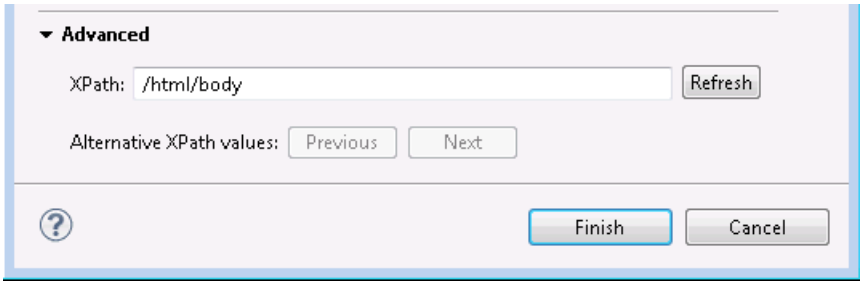
   This dialog box may slightly vary according to the type of element captured such as text fields, text areas, passwords, radio buttons, check boxes and drop-down lists, buttons, and hyperlinks.

5. Defining the captured element:

Provide a suitable name for the element. By default if the HTML element has a name, then this is used by default. If it does not have a name, then the ID is used, and if it does not have an ID then the default is <element type>_<number> (for images, the alt attribute is used). Determine whether this element is to be used as an output/input (depending on the element type) in the procedure (by default the element is selected to be used as the input/output of the procedure, the only exception being when defining a list of elements, when for some types, this check box is not selected by default). The Content area provides a textual display of the captured element (this area does not appear in the Button element).

6. Understanding the element's XPath:

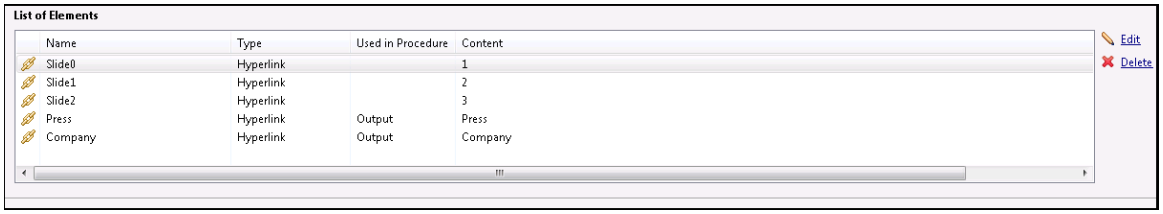
   We use the XPath, the XML Path Language, to locate the element within the Web Page. The wizard tries to suggest the most suitable XPath and in most cases you are not required to make changes. The Advanced section enables you to view the current XPath and if desired change it.



   This may be necessary, for example, to ensure a more robust XPath or if the element that is highlighted in the browser is not the desired element. You can scroll through a list of suggested alternative XPaths (using the Previous and Next buttons) or edit and change the XPath yourself. When making changes to the XPath, it is recommended to click on the Refresh button to ensure that the XPath is valid (for more information regarding XPaths, refer to What is an XPath?). This may also change the contents of the Content area.

7. Reviewing the list of captured elements:

   Once you have completed defining the captured element, the element will appear in the **List of Elements** table beneath the browser area. This list includes all the elements which have been selected on the page currently displayed. Captured elements are hihglighted in dark blue. When you select an element within the List of Elements table, the element will be highlighted in orange.



Repeat steps a-f to select and capture additional elements.

8 **To edit a captured element:**

Either click on the relevant element within the Element List table and then click on Edit, or click on the element within the browser. The *Edit Element* dialog box is displayed and you are able to make the required changes.

9 **To delete a captured element:**

Click on the relevant element within the Element List table and then click on Delete.

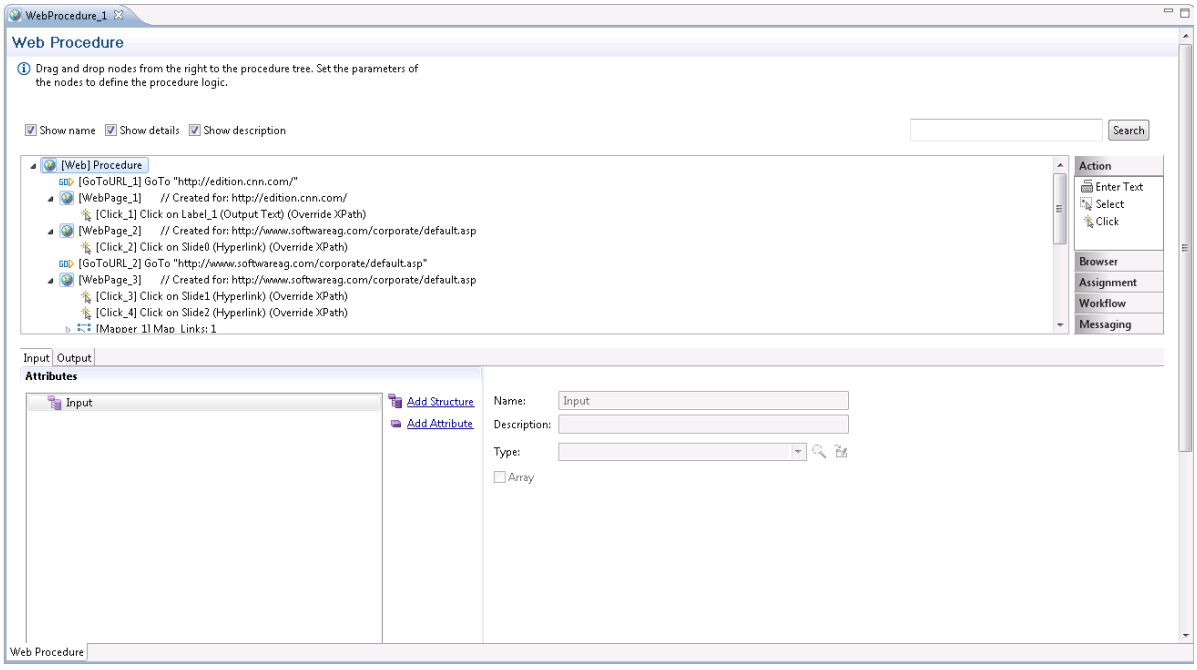
10 **To exit the Selection Mode and continue navigating within the browser**

Click on the Selection Mode button.

11 **To navigate to a different page:**

Either enter the new address in the address field or click on the desired link.

12 To complete the recording and save the entity, click on the **Finish** button. The entity is opened for editing in the Editor area.



Refer to the Web Procedure Solution for further information.

Running the Web Procedure

The Web Procedure can be used to expose web services (Procedure Groups are required for this) or executed from within another procedure. To check that the procedure functions as expected, after you have completed **recording the Web Procedure** you can run the procedure from within the Designer.

To run the procedure, within the Editor, right-click on the relevant procedure and select **Run**. The execution flow is displayed in the Console and the main steps are written to the server log. Sometimes, the procedure requires entering values for input parameters. In such a case a pop-up appears.

For example:

```
<In>

<password></password>

<reconnect>true</reconnect>

<user>demo5</user>

</In>
```

When running the procedure for the first time, the values displayed within the tags are default values that were recorded when creating the Web Procedure. When running the procedure additional times, the last entered values are used.

Sometimes the procedure may fail to run. The Troubleshooting section may provide some useful tips.

Editing the Web Procedure

- Adding a GoTo Node, to Navigate to a URL
- Adding Conditional Logic to the Web Procedure
- Adding a New Element to an Existing Web Page
- Handling Elements whose XPath has Changed
- Working with Dynamic XPaths
- Checking whether an Element Exists within a Web Page
- Retrieving an Element's Value

When completing the process of creating a new Web Procedure, it is opened in the Editor. As with other entities, it is possible to open the Web Procedure and edit it at a later stage. In the Editor you can refine the Web procedure, add logic, add, edit or delete elements and maintain the Web Page structure.

Adding a GoTo Node, to Navigate to a URL

After the procedure is created, the original site may have changed and another page may have been added making your recording out of date. It is also possible that you simply want to add navigation to an additional URL. This can be done by adding a GOTO URL node to the procedure, and defining there the relevant URL. Refer to Working with Procedure Nodes and Web Procedure Nodes.

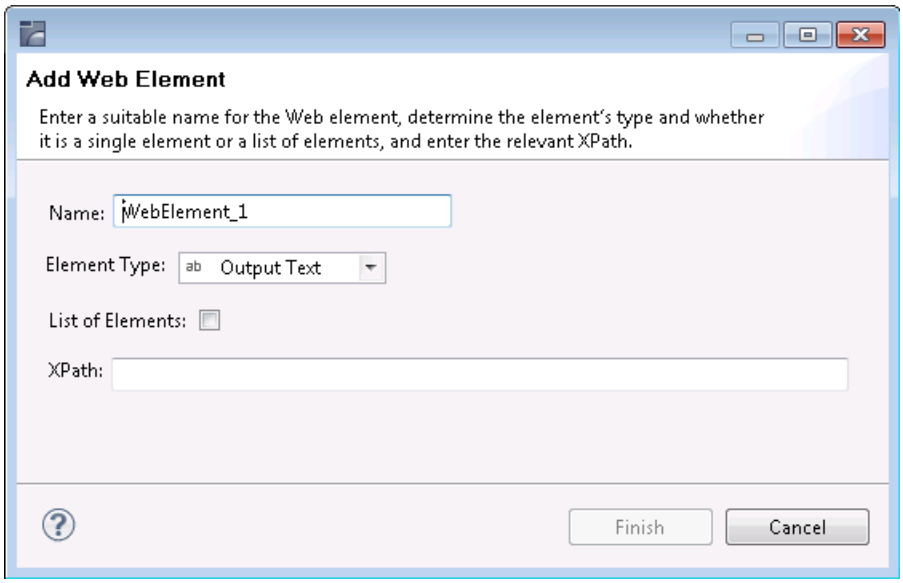
Adding Conditional Logic to the Web Procedure

When recording the procedure, it is not possible to record a conditional scenario such as if a specific field has a specific value then the procedure should then navigate to a specific web page. Using the procedure nodes, actions and expressions to determine the condition, and then the GOTO URL navigation node to navigate to a different URL, you can add such logic into your procedure. Refer to Working with Procedure Nodes, General Procedure Nodes and Web Procedure Nodes.

Adding a New Element to an Existing Web Page

You may find that you need to add an additional element to the Web Procedure. This can be for a number of reasons such as you may realize that you did not capture all the elements that you require or that the Web Page you originally recorded has changed and now you would like to add new elements that did not previously exist.

To add an element, select a Web Page node in the Procedure tree, and in the node details area, where the structure of the Web page and its elements are displayed, right click on the root node of the Web page and select **Add Web Element**. The *Add Web Element* dialog box is displayed.



Enter a name for the element, select the relevant type and enter the XPath. Click **Finish**. The new element will be added to the Web Page schema and can then be used in child nodes.

Handling Elements whose XPath has Changed

A Web site which undergoes changes can cause the XPath of an element to change. As you would still like the Web procedure that you originally defined to be valid, you need to correct the XPath of the element to suit the new one:

1. **Add a new element** to the Web page.
2. In all the Mappers and actions, change the references to refer to the new element instead of to the old, outdated, element.
3. Delete the old element.

Working with Dynamic XPaths

A Web element that is dependent on a user's action or input in runtime, requires an XPath that can change dynamically to suit this. In such cases, after recording the Web procedure and capturing the relevant element, change the XPath (in the procedure editor) by clicking on the Override option where this element is used (in one of the following Web Procedure Nodes: Enter Text action node, Select action node or Click action node) and defining the XPath using logic that will provide the required flexibility in runtime.

When mapping such an element as the output of the procedure, you may require using the Web Element Content expression.

Checking whether an Element Exists within a Web Page

It is possible to check whether an element exists within a specific web page using the Test Web Element Exists expression within the Web Page context.

To check whether a Web element exists:

1 Select Test Web Element Exists from the list of Boolean expressions.

2 Click Is/Is not to define the condition.

3 Click the XPath to open the *Free Text* dialog box and enter the XPath (you can copy-paste the XPath of an existing element by right-clicking an element and selecting Copy XPath to Clipboard).

Retrieving an Element's Value

As part of the Web procedure's logic, you may want to use or check the value of a certain element. This is done using the Web Element Content expression.

To retrieve the value of an element:

1 Select Web Helper>Web Element Content expression.

2 Select the XPath from which to retrieve the content.

Defining Procedure Inputs and Outputs

Procedure inputs and outputs are defined when selecting the procedure root node. They may be used in any node of the procedure.

To define inputs and outputs

1 Within the Procedure root node of a procedure which you created, select the Input or Output tab as required.

2 Add attributes (Refer to Procedure Input and Output Attribute Types) and/or structures (Refer to **Data Structures**).

Working with Procedure Nodes

Nodes are used in Procedures and determine how the procedure will behave. Procedures consist of a number of nodes. These nodes are defined by the user to perform logical operations and are arranged in the order that these operations must be executed.

Note: It is recommended to maximize the Editor screen when defining a procedure.

The nodes in the procedure can display different information regarding the nodes by clicking on the different check boxes (**Show name, Show details, Show description**).

Searching within nodes

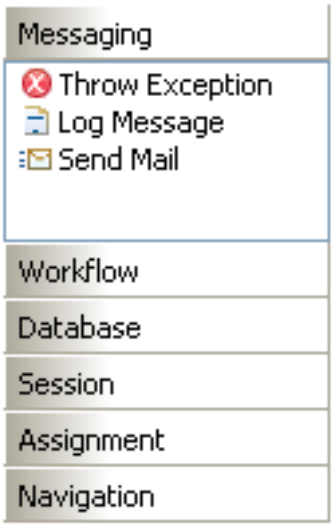
The Search field enables searching for a key word within the procedure nodes. The search mechanism is not case sensitive. The text is searched for within the description, details or name of the nodes. The nodes which contain the text will be selected and expanded in the tree, while other nodes will be collapsed.

Saving the Procedure Tree as an Image

An image of the Procedure tree can be saved as a PNG file, by right-clicking on the Procedure root node, and selecting **Save as Image...**.

To add a node

1 In the Procedure editor, select a node from the list of nodes.



2 Drag and drop the node to the desired location within the procedure.

3 Edit the node.

To delete a node

1 Select the node within the Procedure.

2 Right-click on the node and select **Delete**.

Moving a node within the Procedure

- A node can be moved within a procedure by simply dragging and dropping it in the desired location.

To add/edit an expression

- Expressions are used in procedure nodes. To edit an expression, right-click and select from the list.

Using the Mapper to Map Source Elements to Target Elements

The Mapper tool enables mapping source elements to target output elements. The left-panel contains the entire source data elements that are available at the current flow scope depending on the node type (these element may also be expressions). The right-panel contains the potential target elements depending on the flow scope and the node type. Using the drag-and-drop operation, it is possible to define that when the Mapper is executed, the value/s in the source element will be copied to the target element. These definitions are indicated by lines, linking between the source element and the target element, and also are listed under the relevant node in the procedure nodes tree.

Note: When mapping a source element (e.g inputAttribute in the screen shot below) to a target element (e.g helpAttribute), and in the same mapper mapping the same element (helpAttribute) to another target element (outputAttribute), as the mapper does not enable defining the order in which the elements will be mapped, this can cause a conflict. Therefore it is recommended to perform such a mapping using two separate mappers.

The procedure nodes that enable using the Mapper tool:

- Create Mappings
- Group Arrays (the target will include objects relevant to the specific Group Array node)
- Execute Procedure (lists only the inputs relevant to the specific procedure)
- Flow Procedure nodes:
  - Create Emulation Session (lists only the inputs relevant to the specific procedure)
  - Create DbSession (lists only the inputs relevant to the specific procedure)
  - DbSelect (lists the dynamic inputs. Refer to Creating a dbSelect node)
  - DbExecute- relevant for Flow Procedures only.
  - RollBack (lists only the inputs relevant to the specific procedure)
  - Commit (lists only the inputs relevant to the specific procedure)
  - Create RPC Session- relevant to Flow Procedures only.
  - End Session (lists only the inputs relevant to the specific procedure)
- Path nodes:

- Step (input to source screen\screen group, output from target screen\screen group)
- Path (lists only the inputs relevant to the specific path)
- Screen Mapper

- The expressions that enable using the Mapper tool:
  - Free Text (using tokens)
  - Execute Procedure
  - Host Keys (using tokens)

There are a number of different types of source and target elements that can be mapped:

- Mapping a Simple Type Element to a Simple Type Element
- Mapping an Array Type Element to an Array Type Element
- Mapping a Simple Type Element to an Array Type Element
- Mapping an Array Type Element to a Simple Type Element
- Mapping a Structure Type Element to a Structure Type Element
- Mapping a Number of Simple Elements to a Structure Type Element

Mapping a Simple Type Element to a Simple Type Element

Drag the element in the source frame of the mapper to the relevant element in the target frame. When the Mapping is executed, the source element will be copied to the target element.

Mapping an Array Type Element to an Array Type Element

When mapping an array to an array, it is possible to copy all the source data in an array element to the target element, or it is possible to copy only some of the data to a certain place in the target array.

To copy an array element to an array element

1. Drag the element in the source frame of the mapper to the relevant element in the target frame. When the Mapper is executed, the source element will be copied to the target element. A line will be displayed between the two, indicating that the value from the source element will be placed in the target element.
2. By default, the data is appended to the existing data. This is indicated by a + sign which appears on the line connecting between the source element to the target element. If the + sign does not appear, double-click on the middle area of the mapper (between the area of the source data and the area of the target data) to display the *Link Properties* dialog box. Select the Append check box and click OK.

To copy an array element or part of an array element using an index

1 To define that all or some of the source array values will be placed in a particular place in the target array, drag a line from the element in the source frame of the mapper to the element in the target frame. A line will be displayed between the two, indicating that the value from the source element will be placed in the target element.

2 Right-click on the middle area of the line to display the Link Properties dialog box.

3 To overwrite existing data, ensure that the Append check box is not selected.

4 To copy the data which is from a certain index in the array, right-click and define the Source Index.

5 To determine that the data will be placed in the middle of the array starting from a certain index, right-click and define the Target Index.

6 When you do not need the entire source array but only a specific number of values, define the number of values in the Count field.

7 Click OK to close the dialog box.

Caution: Ensure, that in the Link Properties dialog box, either the Append check box is selected or the Source or/and Target Index have values, otherwise the procedure may not function as you intended and may also cause performance problems.

Mapping a Simple Type Element to an Array Type Element

The simple element can be added to the array in two ways: by appending the value to the existing array or by placing the value in a specific place in the array.

To append a simple type element to an array

1 Drag the element in the source frame of the mapper to the relevant element in the target frame. When the Mapper is executed, the source element will be copied to the target element. A line will be displayed between the two, indicating that the value from the source element will be placed in the target element.

2 Right-click on the middle area of the mapper (between the area of the source data and the area of the target data) to display the Link Properties dialog box. In the Target Index field right-click on <exp> to set the index value where the element will be placed.

3 Click OK to close the dialog box.

Caution: Ensure that if you remove the select from the Append check box, you define the Target Index, otherwise the value of the simple type element will be placed in the [0] index.

Mapping an Array Type Element to a Simple Type Element

To map a value from an array type element to a simple type element, it is necessary to define the index of the relevant value in the array. If you do not define this index, the [0] index will be used.

To place an array type element value in a simple type element

1. Drag the element in the source frame of the mapper to the relevant element in the target frame. When the Mapper is executed, the source element will be copied to the target element. A line will be displayed between the two, indicating that the value from the source element will be placed in the target element.
2. Right-click on the middle area of the mapper (between the area of the source data and the area of the target data) to display the Link Properties dialog box. In the Source Index field right-click on <exp> to set a value for the index.
3. Click OK to close the dialog box.

Mapping a Structure Type Element to a Structure Type Element

The process of mapping the attributes of a structured element to the attributes of a structured element is similar to mapping simple and array elements (refer to Mapping Source Elements to Target Elements). When mapping an element in a structured element to an element in a structured element, map the relevant internal elements and do not map the structured root element.

Caution: Mapping the structure root to the structure root will only map elements of the source structure to the elements of the target structure it they have the same name. Always verify that the correct links are made when mapping different types of structures.

Mapping a Number of Simple Elements to a Structure Type Element

It is possible to map a number of simple elements to a structure.

To map a number of simple elements to a data structure type element

- Select more than one simple element in the source frame of the mapper using CTRL or SHIFT. Drag the selected elements to the relevant data structure in the target frame. When the Mapper is executed, the source element will be copied to the target element within the data structure. A line will be displayed between the two, indicating that the value from the source element will be placed in the target element.

Program Procedures

To understand the Program feature in ApplinX, we first need to overview the concept of Remote Procedure Call (RPC). RPC is a description for APIs that permit distributed applications to interact with each other across the network. As part of its host integration capabilities, the ApplinX server implements RPC calls to execute host programs and routines, thus adding another type of interaction with hosts in addition to screens and screen data.

This Wizard guides you through a sequence of instructions to import a program object and corresponding field elements. Currently, the Wizard supports the import of program objects for COBOL and RPG programs for AS/400 hosts.

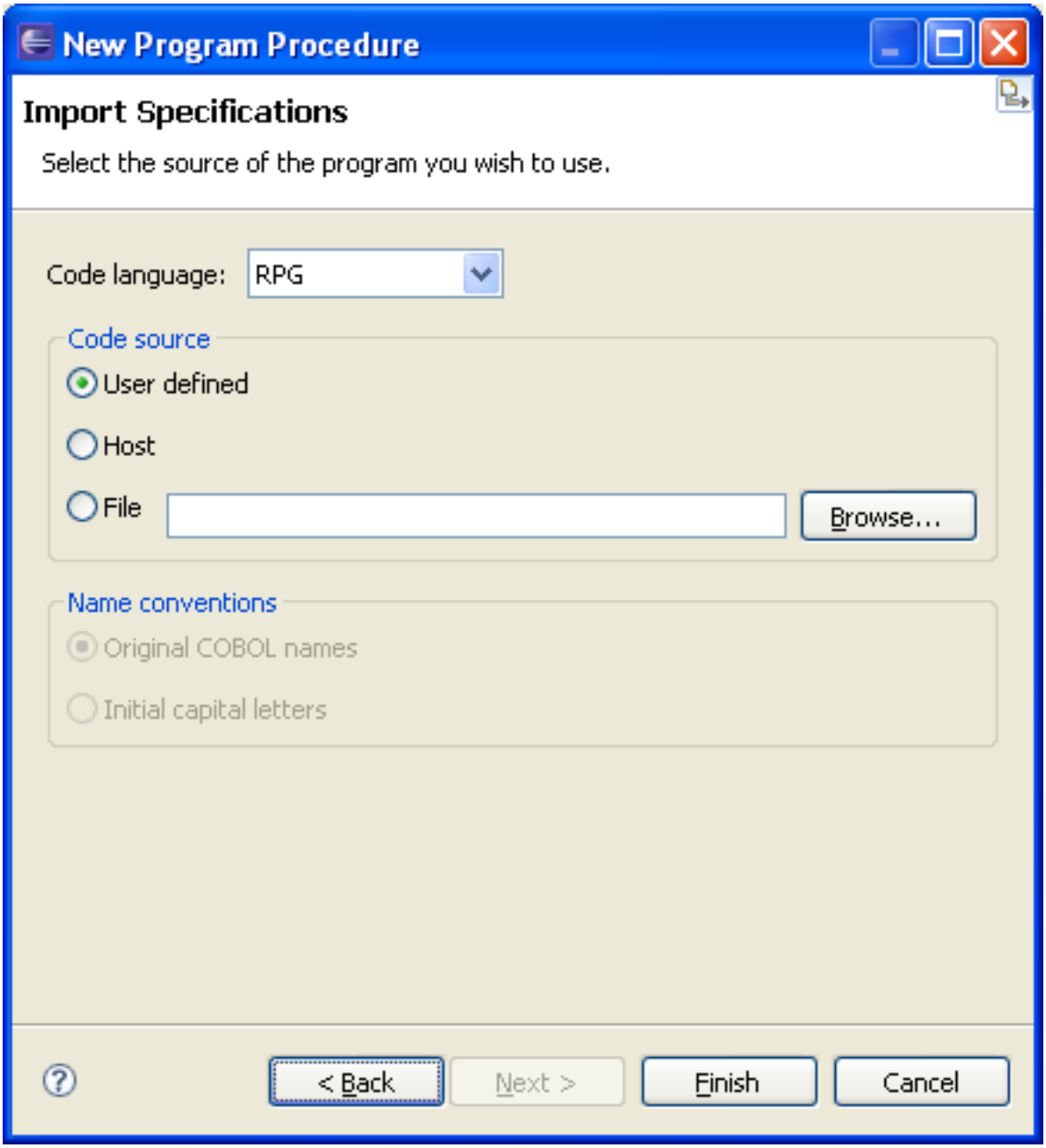
To create a new Program Procedure

1 Select the relevant application or the Root node of the application.

2 In the ApplinX Explorer tool bar, click on the arrow to the right of the **Create new entity** icon and select **New Program Procedure**. The New Program Procedure wizard is displayed.

3 Enter a name and description (optional) and click **Next**.

4 The *Import Specifications* screen is displayed.



5 From the Code language drop-down list select COBOL or RPG.

6 Specify the code source:

- User defined: Enables you to define a new program.
- Host: Enables you to import a program from a host.
- File: Enables you to import a program from a file. Specify the code source file.

7 For more convenient use, ApplinX can rename the parameters automatically for you. Advanced users (AS/400 hosts only): When RPG is selected, ApplinX can retrieve the Program's source

file directly from the AS/400 machine. Click the Code retriever settings button to display the Program - Advanced dialog. Specify the Host Library, File, Member and Line Length (default is 100). Enter the host's address or select a previously defined host from the drop-down list. Enter the Host's User name and Password, otherwise the default values as set in the host configuration are used. Click **OK**. Click **Next.**

Defining Program Procedure Inputs and Outputs

- Double-click on the relevant Program Procedure from within the Repository. The Editor area displays the Procedure Program. Add/edit parameters. Refer to the parameter details.

External Web Services

The External Web Service feature enables creating procedures based on a WSDL which describes a Web service within the organization. An External Web Service Procedure can be invoked, either by Flow, Path or Web Procedures.

Refer to External Web Services Referencesection.

Note: This feature is only available when working with a SOA license.

Current limitations:

ApplinX supports only DOC/Literal (wrapped and bare) WSDLs.

ApplinX supports the following XSD types (refer to http://www.w3.org/2001/XMLSchema) and any ComplexType which uses them:

- String
- Boolean
- Double
- Float
- Int
- Integer
- Long
- Date (The format can only be 2001-07-04T12:08:56.235.
- dateTime

External Web Services currently do not support the following services:

- Services that contain Web methods that use data types comprised of ANY tags.

For example:

```
<s:element minOccurs="0" maxOccurs="1"
  name="GetHolidaysForMonthResult">
          <s:complexType>
            <s:sequence>
              <s:element ref="s:schema" />
              <s:any />
            </s:sequence>
          </s:complexType>
         </s:element>
```

- Services that contain Web methods that use data types comprised of complex data types defined within the parent data type definition.

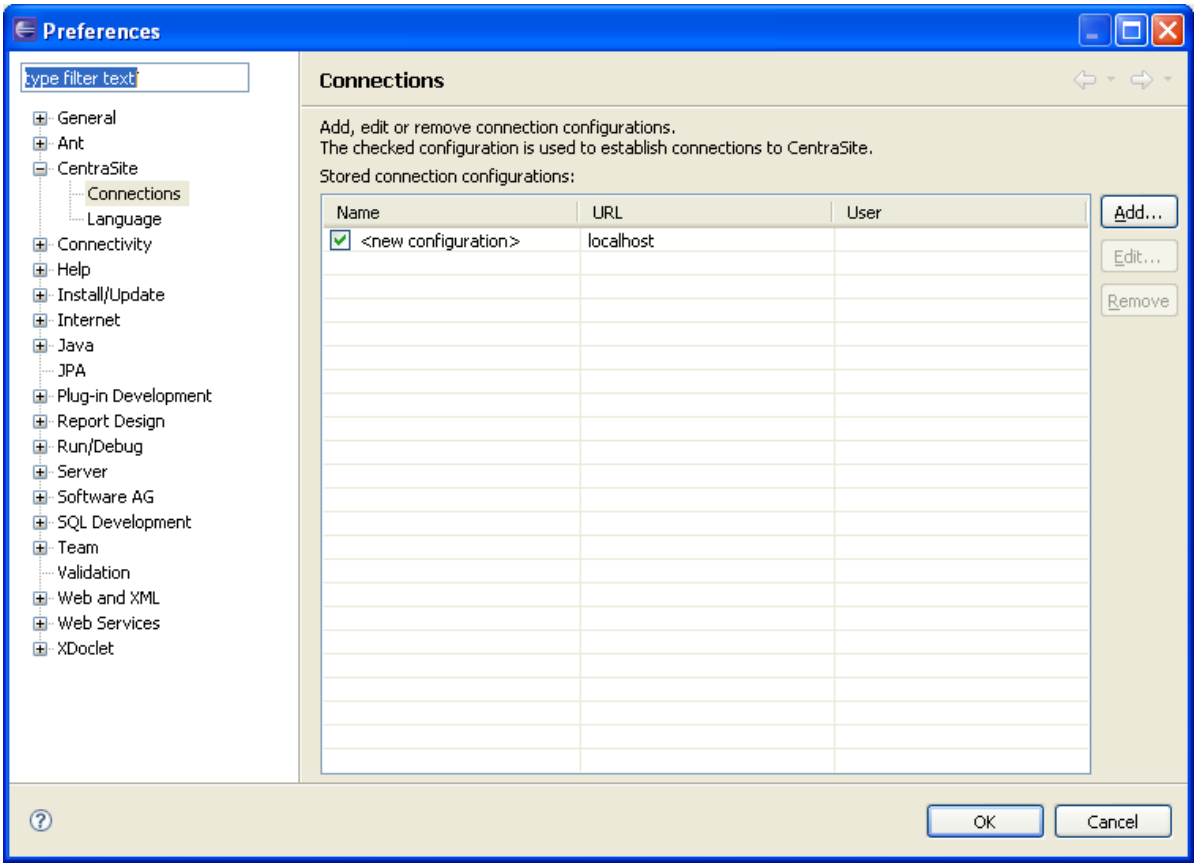
  For example:

```
<s:element name="GetHolidaysForDateRangeResponse">
     <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
           name="GetHolidaysForDateRangeResult">
            <s:complexType>
              <s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="year" type="s:int" />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
       </s:complexType>
      </s:element>
```

Creating an External Web Service

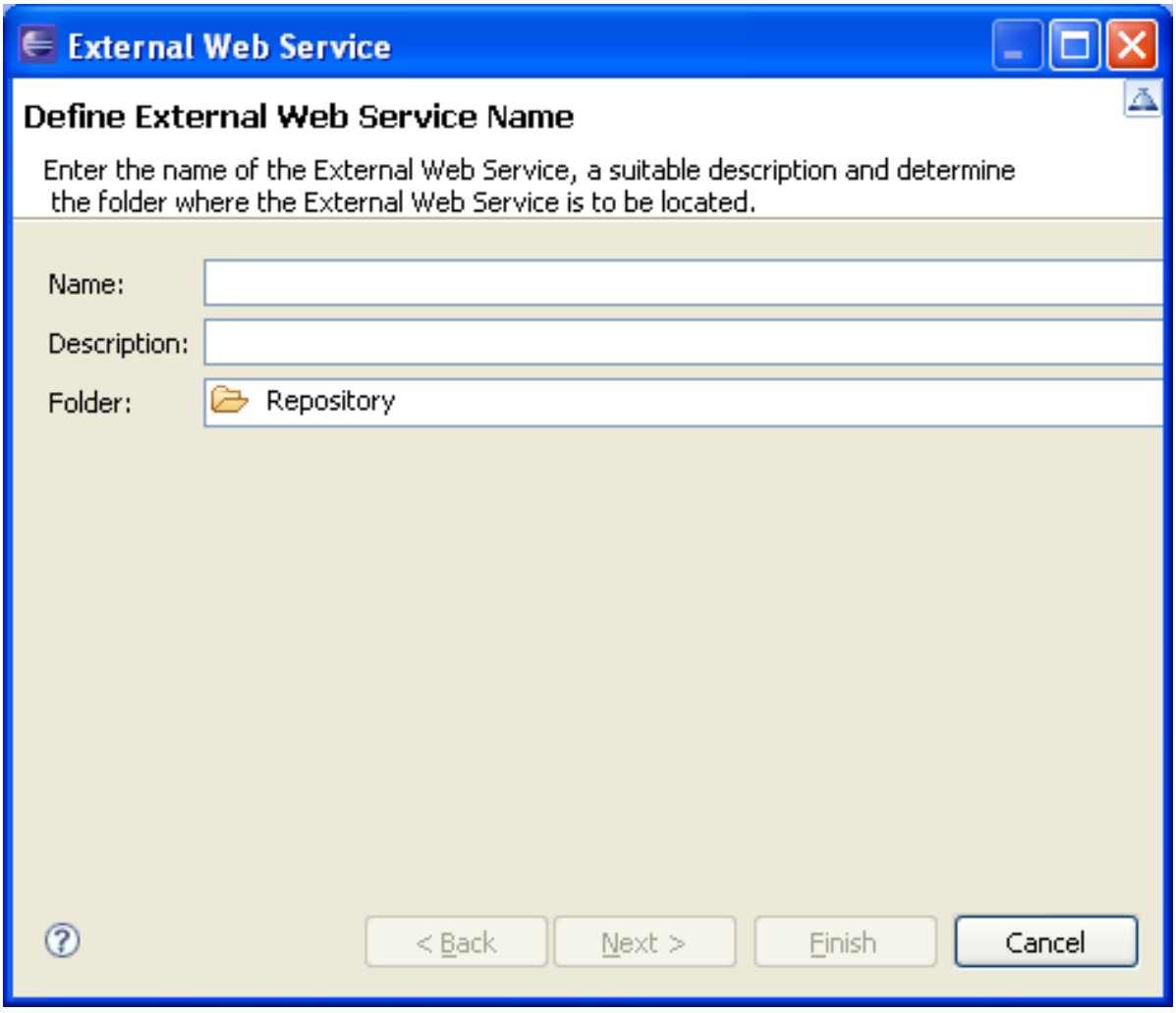
1 Select the relevant application.

2 In the **File** menu select **New>Entity>External Web Service**. If you have the CentraSite plug-in, but have not configured the CentraSite connection, the Eclipse Preferences dialog box is displayed requiring you to configure the connection. This is not required when you want to use a WSDL URL.



If you wish to use a Web Service that is registered in CentraSite, ensure that you have set a connection configuration in the Eclipse Properties dialog box and that you supplied a valid user name and password (these will be used when attempting to retrieve the WSDL file from CentraSite). It is recommended to test the connection to CentraSite (Test button), after configuring the connection.

3 The External Web Service wizard is displayed.



4 Enter a name and click Next.

5 The Web Service Source screen is displayed.

6 Select whether to use a WSDL URL (if you don't have CentraSite or if the Web Service is not registered in CentraSite) or CentraSite.

7 When relevant, enter the WSDL URL or select a file name.

When selecting CentraSite, you will be required to enter the user name and password to be used when retrieving the WSDL from CentraSite (the default values are taken from the CentraSite preferences). Then, select one of the web services (in the Result section). Refer to the CentraSite documentation on Empower for further information regarding this dialog box.

When entering a WSDL URL, you may be required to provide a user name and password.

8 Click **Finish**. The External Web Service is displayed in the Editor area. The General tab displays the basic information about the web service and the Structure tab displays the contents of the Web service.

Note: It is possible to change the target server. This can typically be used when changing the environment, such as from developing and testing environment to the production environment (which may be on a different server than the developing and testing server), as when changing the environment, the URL will be broken, causing the External Web Services not to work. This is done by clicking on the relevant button in the General tab and then selecting the new address (IPv4 and IPv6 address formats are supported) and port. When necessary, you can also select to update the WSDL URL accordingly. These changes can be applied just to the current Web Service, or to all External Web Services using the same target server (when selecting this option, the list of updated services appears in the server log).

Debugging Procedures

After creating a procedure, use the ApplinX Debugger tool to observe the run-time behavior of your procedure and determine the location of logical errors. With the debugger, you can control the execution of your program by setting breakpoints, stepping through your code, running the code to the cursor position and examining the contents of variables. Refer to the Eclipse help for further details on the Debugger.

Running the Debugger

Debugging procedures are based on the given capabilities of the Eclipse based debug tools. You can debug "step-by-step", using the "step-over" action (F6). You can achieve "Run to Cursor" by setting the breakpoints in the proper place and using the "run/resume" action (F8).

To debug a procedure

1 Right-click on the relevant procedure and select Debug or Run.

Note: The first time a Path Procedure that uses a Connection Pool is debugged, the Session Selection dialog box is displayed, requiring you to select the relevant session to use when debugging this Path Procedure.

2 Sometimes, the procedure requires entering values for input parameters. In such a case a pop-up appears. For example:

```
<In>
<password></password>
<reconnect>true</reconnect>
<user>demo5</user>
</In>
```

The values displayed within the tags are default values set by the user.

- When deleting a text value and/or not entering a value (e.g. `<password></password>`), an empty string is used.
- When deleting a boolean, numeric or date value and/or not entering a value, the ApplinX default is used (ApplinX default values are as follows: for a boolean parameter - false, numeric - 0 and date - current date).
- When removing a whole element (such as deleting `<reconnect>true</reconnect>`), the user defined default is used when one exists, otherwise, the ApplinX default is used (ApplinX default values are as follows: for string parameters - null, boolean - false, numeric - 0 and date - current date).

17 Procedure Groups (used as Service Providers)

A Procedure Group is required when exposing procedures using Web services or the Procedure client. A procedure group consists of a number of procedures.

Creating a Procedure Group

To create a new Procedure Group

1 Select the relevant application.

2 In the **File** menu select **New>Entity>Procedure Group**. The *New Procedure Group wizard* is displayed.

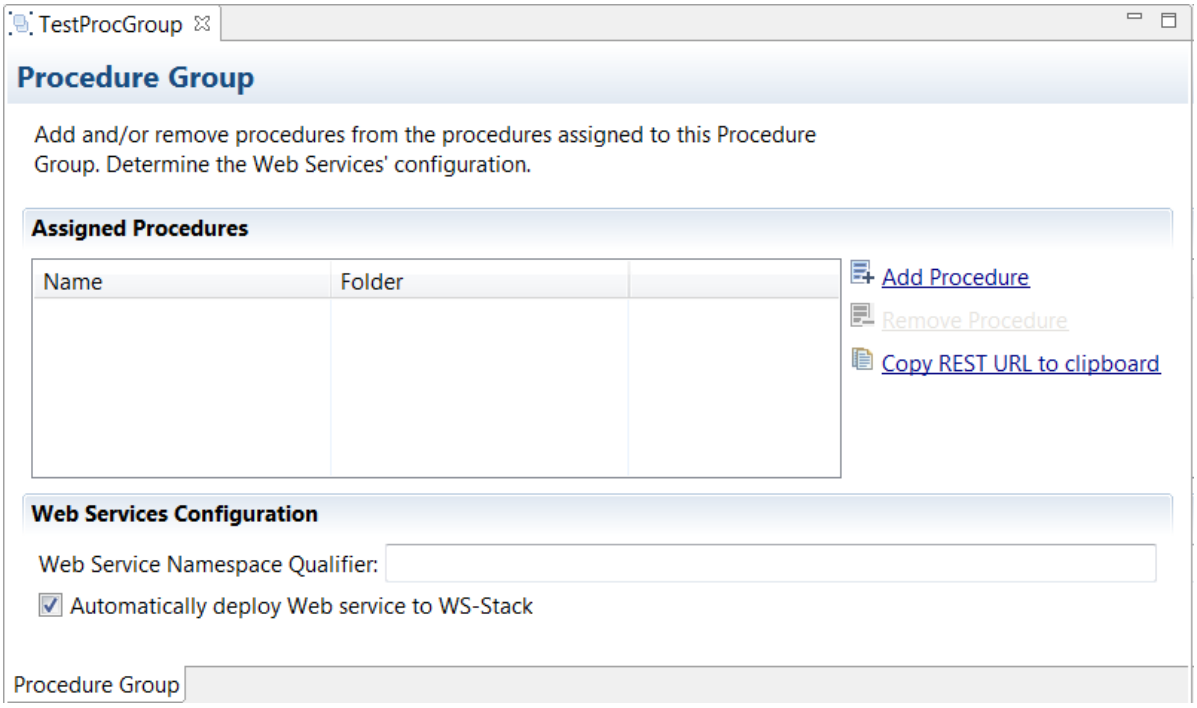
3 Enter a name for the Procedure Group, a suitable description and determine the folder where the screen is to be located. Click **Finish**. You have now created a Procedure Group entity in the repository.

See also: **Procedure Groups' Generated Web Services** and **Deploying a Procedure Group to WS-Stack**

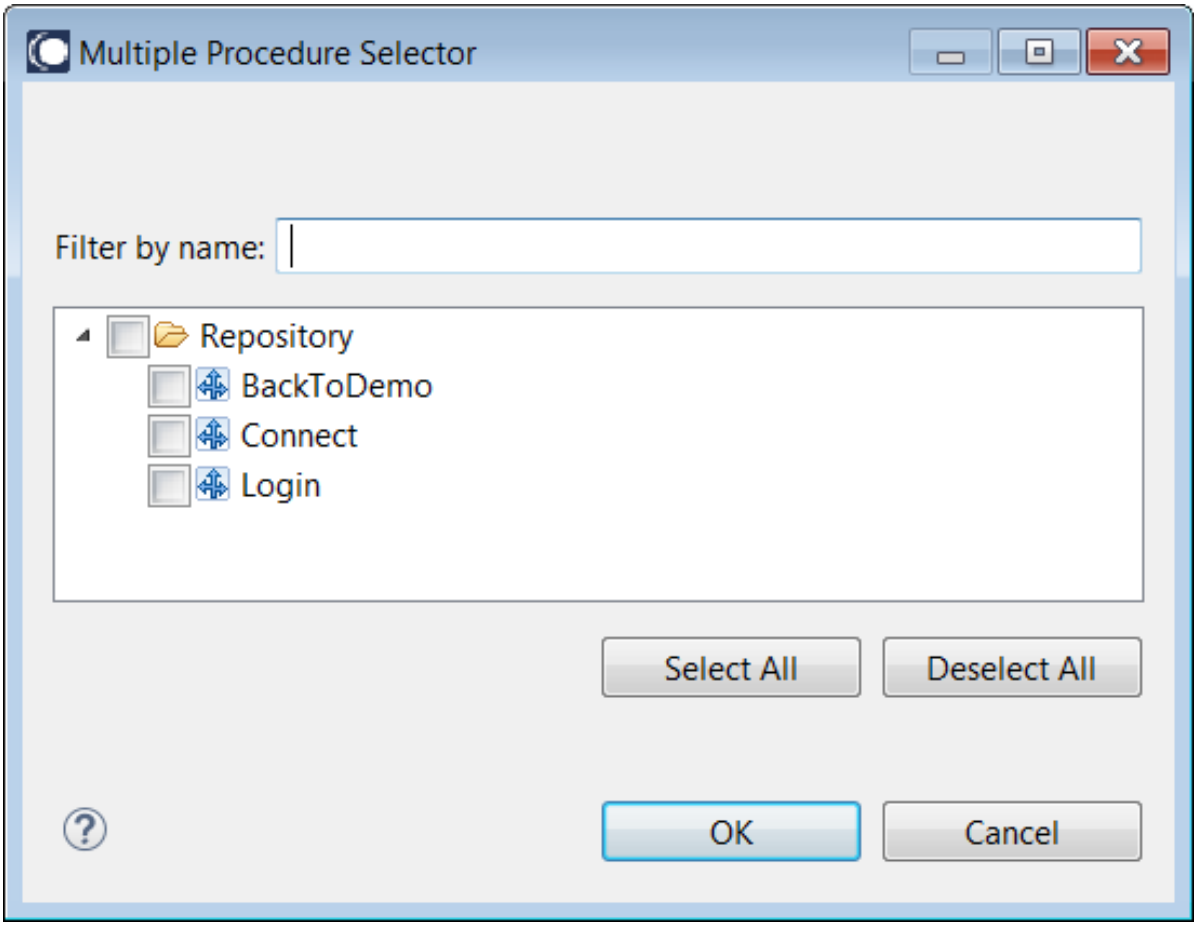
Assigning a Procedure to a Procedure Group

To assign a Procedure to a Procedure Group

1 In the ApplinX Explorer, double-click on a Procedure Group. The Procedure Group will be displayed in the Editor area.



2 Click on the **Add Procedure** link. The *Multiple Procedure Selector* dialog box is displayed.



Select the procedures that you want to assign to this Procedure Group. Click **OK**.

See also: **Procedure Groups' Generated Web Services** and **Deploying a Procedure Group to WS-Stack**

ApplinX as a Web Service Provider

- Procedure Groups' Generated Web Services
- Deploying a Procedure Group to WS-Stack
- Registering a Web Service to CentraSite
- Registering a Web Service to webMethods Integration Server
- Creating or Updating an Adapter Connection
- Creating or Updating a REST Resource

- Integration between ApplinX and WS-Stack

Procedure Groups' Generated Web Services

- Introduction
- Invoking a Procedure from a SOAP Client
- Invoking a Procedure, Using REST API

Introduction

A Procedure is a well-defined business-logic (with input and output arguments). A Procedure Group is a container of Procedures. Procedure Groups are exposed as Web services via WS-Stack. Flow procedures, path procedures, Web procedures, programs and path wrappers that are placed within a procedure group can be exposed as operations within a Web service. ApplinX allows you to connect to the ApplinX server and request information from it, which can be obtained from a legacy system. In other words, you can connect to the ApplinX Server through your browser and ask the server to perform some logic by way of procedures (Web methods) to receive the required information.

Note: The user is sometimes required to enter input values for parameters. In such a case a pop-up appears. For example:

```
<In>
<password></password>
<reconnect>true</reconnect>
<user>demo5</user>
</In>
```

The values displayed within the tags are default values set by the user.

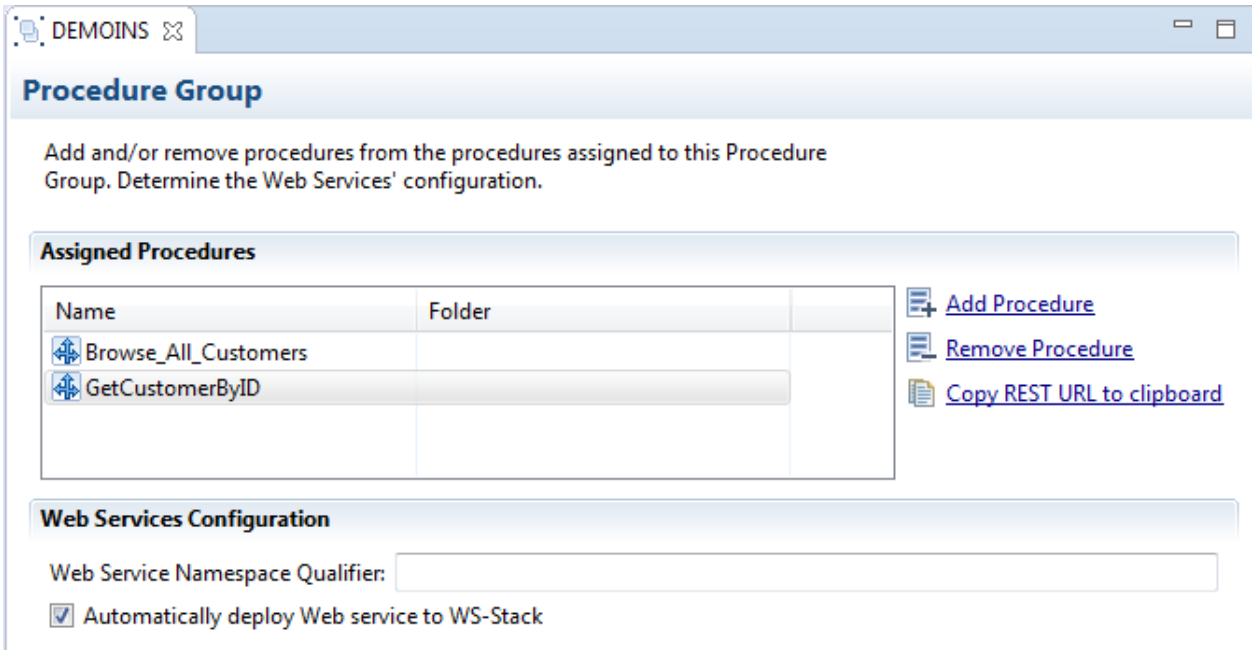
- When deleting a text value and/or not entering a value (e.g. `<password></password>`), an empty string is used.
- When deleting a boolean, numeric or date value and/or not entering a value, the ApplinX default is used (ApplinX default values are as follows: for a boolean parameter - false, numeric - 0 and date - current date).
- When removing a whole element (such as deleting `<reconnect>true</reconnect>`), the user defined default is used when one exists, otherwise, the ApplinX default is used (ApplinX default values are as follows: for a boolean parameter - false, numeric - 0, date - current date and string - empty string).

Invoking a Procedure from a SOAP Client

To invoke a procedure from a SOAP client, right-click on the relevant Procedure Group, and select **Copy WSDL URL to Clipboard** and paste it into your external application (right-clicking on the Procedure Group and selecting **Show WSDL** will display the WSDL file within a browser).

Invoking a Procedure, Using REST API

ApplinX web services can be invoked using REST. From the Procedure Group Entity Editor, select a procedure and choose **Copy REST URL to clipboard**.



The URL is copied to the clipboard, and you can then consume it using any client. Currently only simple GET requests with flat input are supported. Structured or complex input is not supported.

The GET request contains all inputs with their default values. If the input does not have a value, the URL will contain a placeholder < your_value_here >, which you should fill with the required values.

POST requests, used for structured inputs, are currently not supported out of the box, but you can consume structured procedures by creating the request in application/xml format.

The example above creates the following REST URL: *http://localhost:2380/wsstack/services/SOADemo.DEMOINS/GetCustomerByID?Select_Customer_ID=27*. The browser's response is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <tns:GetCustomerByID_Response xmlns:tns="http://tempuri.org/">
    <tns:Addressform />
    <tns:CID>27</tns:CID>
    <tns:City>London</tns:City>
    <tns:Country_Code>GB</tns:Country_Code>
    <tns:Date_of_Birth>1989-04-16</tns:Date_of_Birth>
    <tns:Fax />
    <tns:Firstname>Charles</tns:Firstname>
    <tns:House_Number>12</tns:House_Number>
    <tns:Lastname>Chappell</tns:Lastname>
    <tns:Mobile />
    <tns:Municipal_Code />
    <tns:Nationality_Long>English</tns:Nationality_Long>
    <tns:Nationality_Short>ENG</tns:Nationality_Short>
    <tns:Occupation_Long>Actor</tns:Occupation_Long>
    <tns:Occupation_Short>81177</tns:Occupation_Short>
    <tns:PO_Box />
    <tns:Person_Sex>M</tns:Person_Sex>
    <tns:Person_Type>I</tns:Person_Type>
    <tns:Phone />
    <tns:Status>A</tns:Status>
    <tns:Street>Cromwell Street</tns:Street>
    <tns:Title />
    <tns:Web_Address>cchçweb.uk</tns:Web_Address>
    <tns:ZIP_Code>S6C 3T2</tns:ZIP_Code>
  </tns:GetCustomerByID_Response>
```

Deploying a Procedure Group to WS-Stack

A Procedure Group can be deployed to WS-Stack manually or automatically. Refer to **Integration between ApplinX and WS-Stack** for further details.

Automatic Deployment

To automatically deploy Web services, access the relevant Procedure Group, and select the **Auto deploy Web service to WS-Stack** check box.

Note: When deploying automatically, ensure that the initialization mode "When first accessed" is selected. If this mode is not selected, the Web Services will not be deployed until the application is loaded.

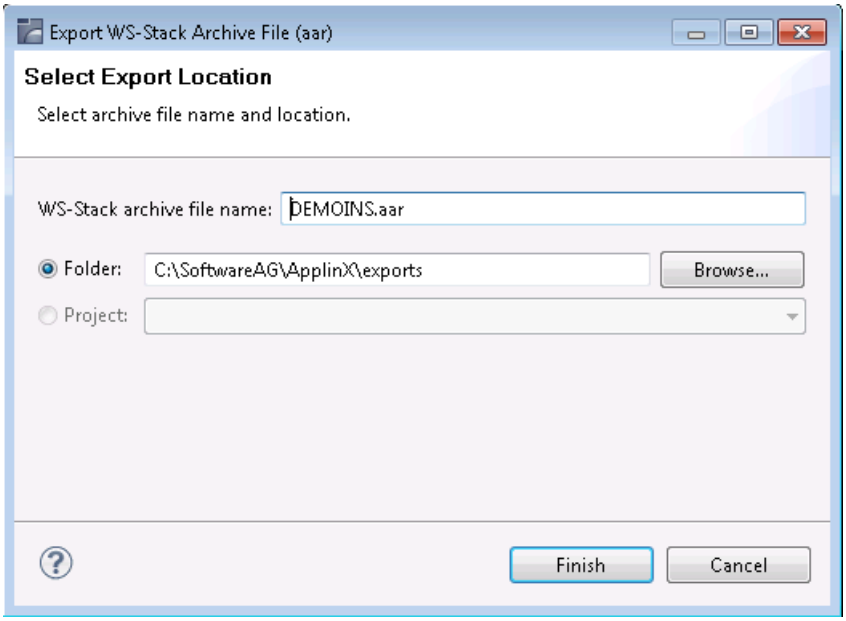
Manual Deployment (relevant only for remote WS-Stack)

Before manually deploying a Procedure Group to WS-Stack, ensure that you have the WS-Stack plug-in for Eclipse installed. If you do not have the plug-in installed, install it from the SAG installer.

To manually deploy a procedure group to WS-Stack

Ensure that in the Procedure Group Editor, the **Auto deploy Web service to WS-Stack** check box is not selected (the following time this Procedure Group is saved, the Web service will be un-deployed).

1 Right-click on the relevant Procedure Group and select **Export WS-Stack Archive...**.



2 Enter a name for the AAR file.

3 Select a folder or an existing project in Eclipse, where the external web service should be placed. Click **Finish.**

The AAR file is created in your Eclipse project (If you selected Folder in the previous step, place the AAR file in the relevant project in Eclipse).

4 Right-click on to deploy. Additional configuration for the web service (such as security) is possible. For further details, refer to the WS-Stack documentation on Empower.

Registering a Web Service to CentraSite

ApplinX procedures can be registered as Web services in CentraSite. CentraSite is an open and standard-based SOA repository that allows customers to manage and govern their SOA environment and to achieve control and transparency across all IT resources related to the SOA, initiative services, policies and processes within the organization.

To register a procedure group

1 **Define the WS-Stack definitions** in ApplinX.

2 Right-click on the relevant Procedure Group and select **Export WS-Stack Archive...**. The Export WS-Stack Archive wizard is displayed.

3 Enter a name for the WS-Stack archive file that is to be created. By default the name is the name of the Procedure Group you selected in Step 2).

4 Select the folder or Java project where the file is to be created (to select a Java Project, ensure that such a project exists).

5 In the Java Perspective of Eclipse, deploy the web service package and then register the web service package (refer to the WS-Stack documentation for further details).

Registering a Web Service to webMethods Integration Server

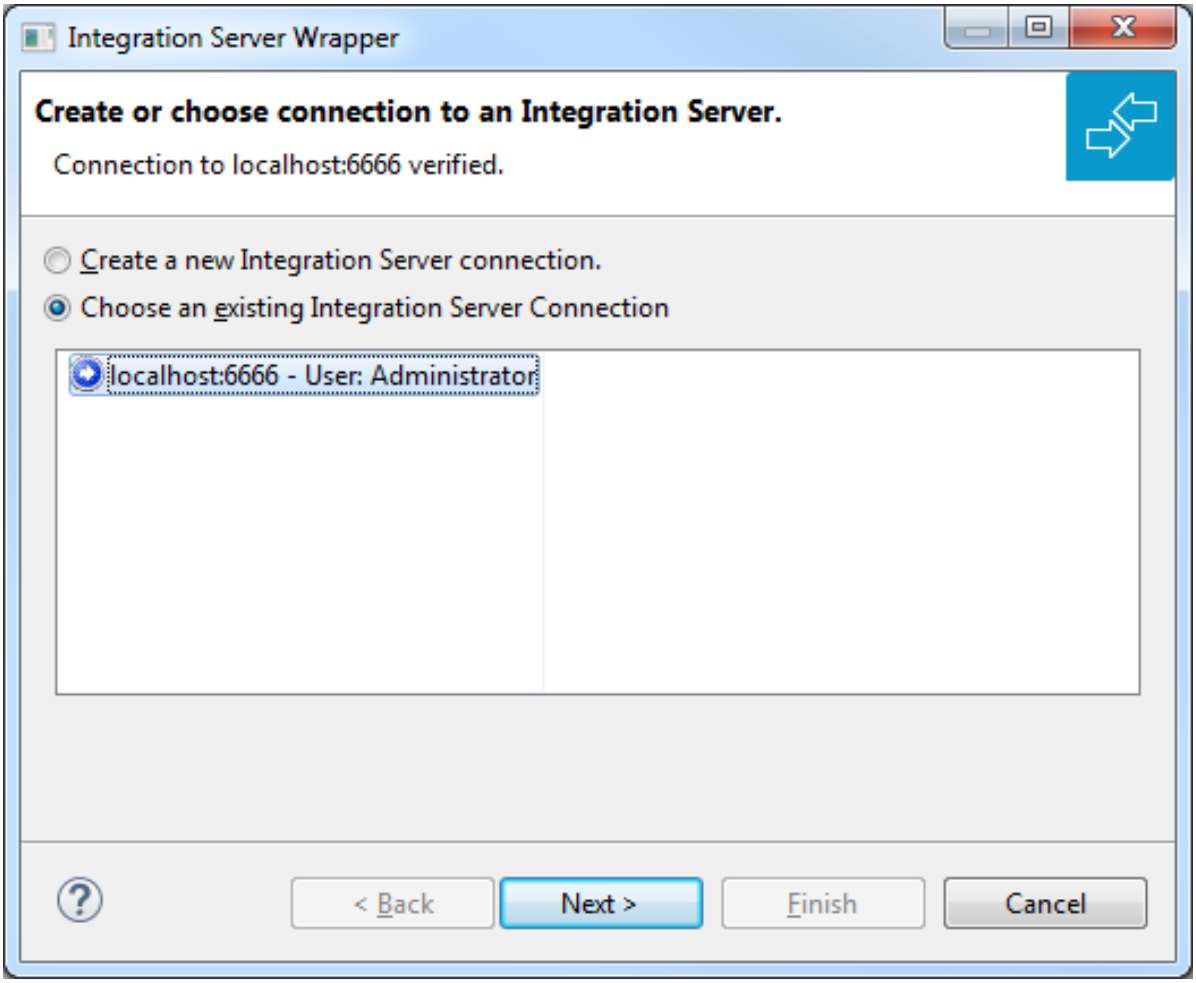
ApplinX procedures can be registered as services in webMethods Integration Server. webMethods Integration Server supports the integration of diverse services, such as mapping data between formats and communication between systems.

Note: This feature is only available when working with a SOA license.

Note: Ensure that you have the EntireX adapter installed (this does not require a special license).

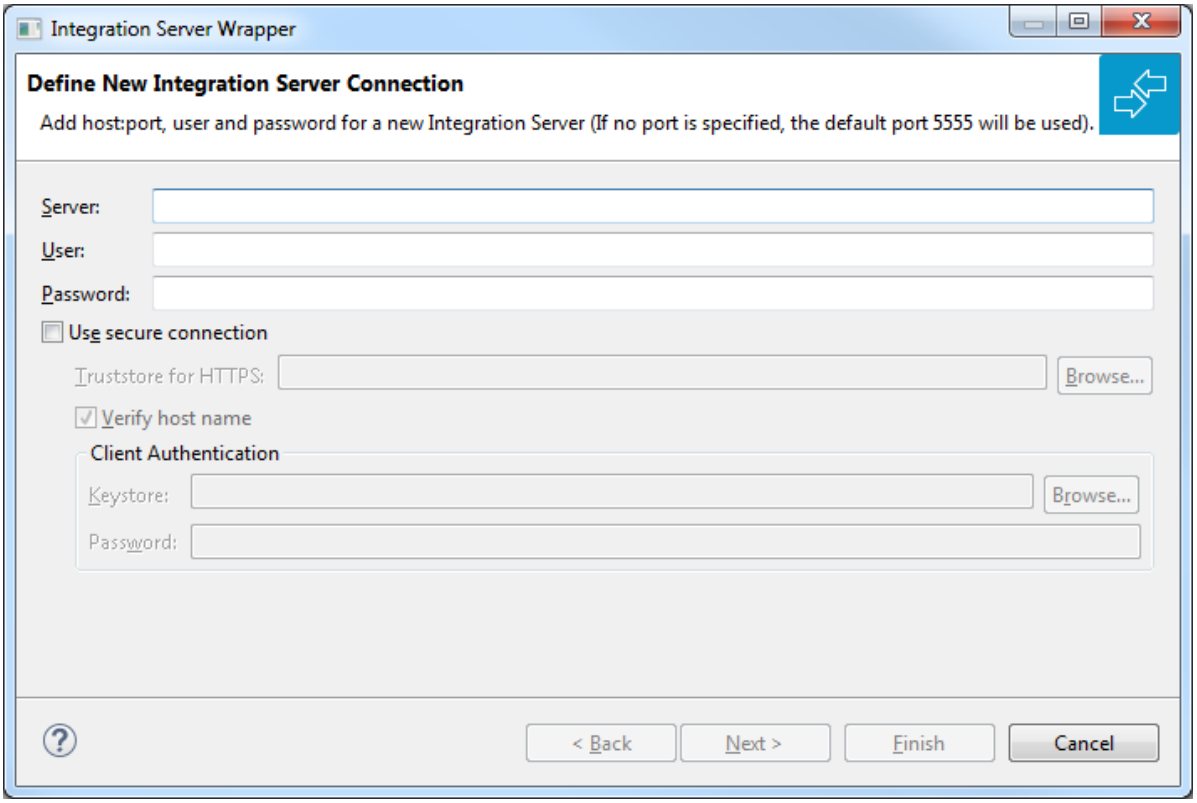
To register a procedure group

1 Right-click on the relevant procedure group and select **Deploy to Integration Server...**. The Integration Server Wrapper wizard is displayed.



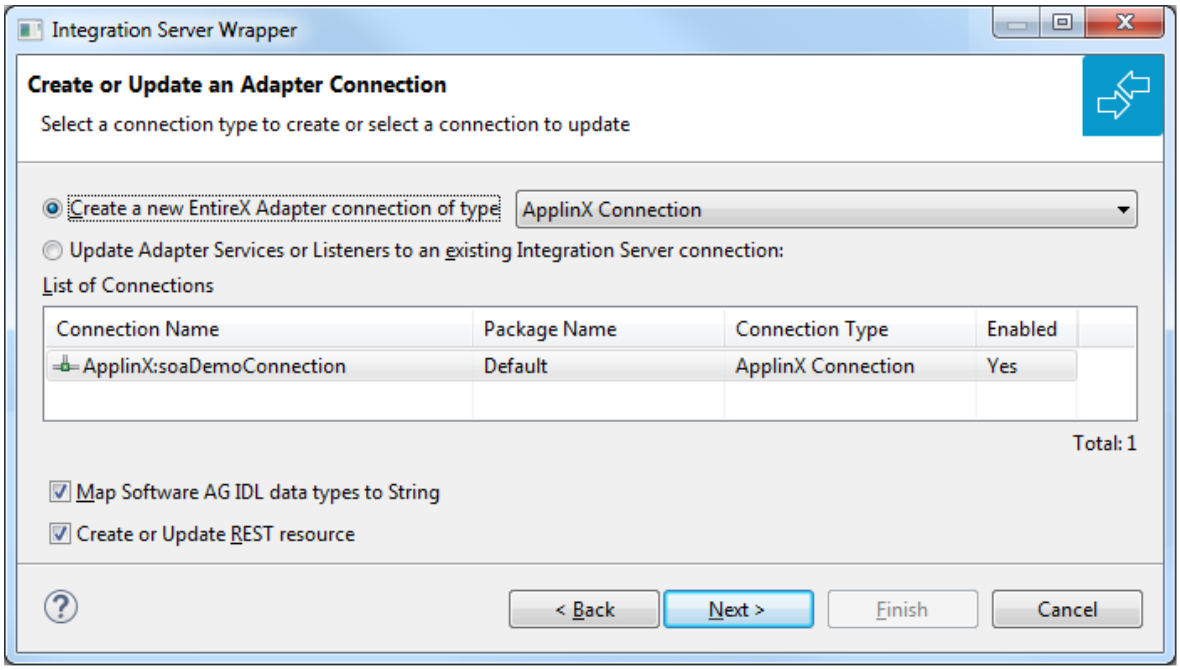
2 Select to create a new Integration Server Connection or choose an existing Integration Server connection. Click **Next.**

3 When selecting to create a new Integration Server connection, you are required to specify the Integration Server's connection settings: the server address (IPv4 and IPv6 address formats are supported), user name and password and secure connection settings.


Integration Server Wrapper
Define New Integration Server Connection
Add host:port, user and password for a new Integration Server (If no port is specified, the default port 5555 will be used).
Server:
User:
Password:
Use secure connection
Truststore for HTTPS:
Browse...
Verify host name
Client Authentication
Keystore:
Browse...
Password:
< Back
Next >
Finish
Cancel

Click **Next.**

Creating or Updating an Adapter Connection



In this step you can either create a new Integration Server connection or update adapter services to an existing Integration Server connection.

To create a new connection

- Select connection type ApplinX connection from the drop down list and press **Next**.

To update an existing connection

1 Select an ApplinX connection from **List of Connections**.

As a result, you are informed on how many adapter services will be created, modified or left unchanged.

The update process can be characterized as follows:

- The metadata is updated for each IDL program.
- An adapter service is created for each new IDL program is created.
- An existing adapter service is updated if it is contained in the IDL file for the update.
- A connection remains unchanged with respect to its type and settings (broker ID, server address, user ID, etc.).

2 Click **Next** or **Finish**.

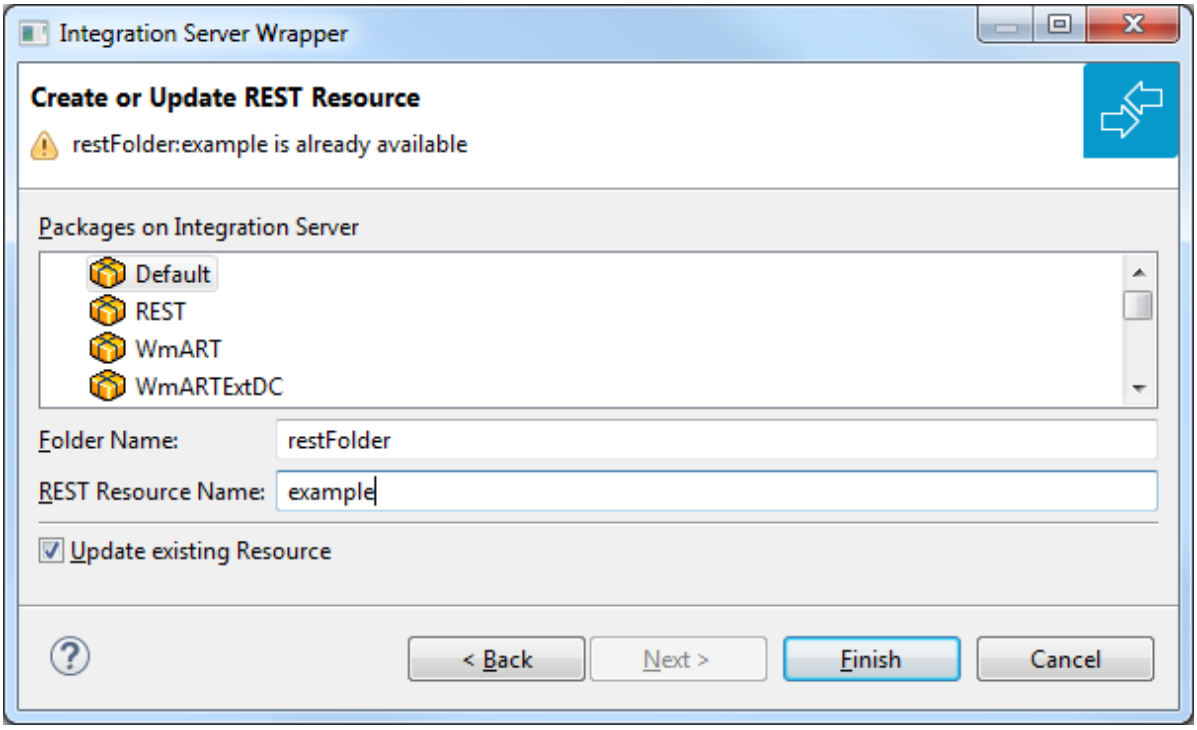
The checkbox **Map Software AG IDL data types to String** specifies how to map IDL data types to Integration Server data types. The default is to map all data types (except binary) to String.

Check **Create or Update REST Resource** if you want to create REST V2 resources for your connection. An additional dialog will appear to define the destination of the resource.

Creating or Updating a REST Resource

To create or update a REST resource

1 Select a package.



2 Define a folder name. If the folder does not exist, it will be created.

3 Define the REST Resource name.

4 Check **Update existing Resource** if you want to add additional services to an existing resource.

See the *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources.

Integration between ApplinX and WS-Stack

What is WS-Stack?

Software AG Web Services Stack (WSS) is a toolkit that provides functionality for execution, configuration and management of web services. It handles the complex process of sending and receiving of web services requests in Software AG products. It allows everyone to do web services while knowing only a few of the details about the web services specifications.

The core part of the WSS runtime is the SOAP engine, based on Apache Axis2. Incoming SOAP requests are processed by this SOAP engine. The SOAP request is given to the SOAP Runtime and sent back to the client as a SOAP response message. If an error occurs a SOAP fault message is sent back to the client.

ApplinX and other products in the webMethods suite use WS-Stack for exposing and deploying web services.

Defining WS-Stack for ApplinX

To configure WS-Stack in ApplinX

For WS-Stack Administration, you must have SMH (System Management Hub) installed. This is provided with the Software AG Installer.

1 Open the *Server Properties* dialog box (right-click on the relevant server in the ApplinX Explorer and select **Properties**).

2 Select whether to work in embedded (default) or in external mode. Use the Embedded mode when WS-Stack uses the same Tomcat as ApplinX and also when working with Web Services created in previous ApplinX versions (Administrative Web Services and Procedure Group generated Web Services). Use the External mode when not using the same Tomcat as ApplinX, for example when working with one WS-Stack Web application for all SAG products or when working with a WS-Stack Web application which is on a different machine.

Embedded Mode

. When working in the Embedded mode, it is not necessary to configure any parameters as the values are configured automatically.

External mode with Web Services Stack that uses a Platform Tomcat Server on the same machine as ApplinX installation

When working with Web Services Stack that uses a Platform Tomcat Server that was installed when installing ApplinX, it is not necessary to configure any parameters as the values are configured automatically. When working with Web Services Stack that uses a Platform Tomcat Server that was installed before installing ApplinX, once ApplinX installation was completed, restart the Platform Tomcat Server by restarting its service - Software AG Tomcat Server Service (sagctp82).

External mode when not working with Web Services Stack that uses a Platform Tomcat Server

In the Server Properties dialog box, configure the WS-Stack External mode parameters: enter the host name, the Tomcat port where WS-Stack Web application is deployed, the URL of the servlet that the WS-Stack uses for deployment tasks and the user name and password used by WS-Stack.

Copy the applinx-wsstack.jar file (can be found in the ApplinX installation wsstack\WEB-INF\lib directory) and the applinx-shared.jar file (can be found in the ApplinX installation shared\lib directory) from the ApplinX installation, to the WS-Stack Web application, WEB-INF\lib directory. Restart the WS-Stack Tomcat.

External mode when installing Web Services Stack that uses a Platform Tomcat Server on a different machine or on the same machine, but after installing ApplinX

In the Server Properties dialog box, configure the WS-Stack External mode parameters: enter the host name, the Tomcat port where WS-Stack Web application is deployed, the URL of the servlet that the WS-Stack uses for deployment tasks and the user name and password used by WS-Stack.

Copy the applinx-wsstack.jar file (can be found in the ApplinX installation wsstack\WEB-INF\lib directory) and the applinx-shared.jar file (can be found in the ApplinX installation shared\lib directory) from the ApplinX installation, to the WS-Stack Web application, <Software AG installation directory>\profiles\CTP\workspace\wsstack\repository\lib. Restart the Platform Tomcat Server by restarting its service - Software AG Tomcat Server Service (sagctp82).

Note: To work in External mode with an application from a previous ApplinX version, manually redefine the Web Services after upgrading the ApplinX application.

Note: Refer to the WS-Stack documentation for further information.

Backwards Compatibility

When working in the Embedded mode, ApplinX supports Web services created in previous ApplinX versions and that have not been edited. Once you make a change in the Web service, and require regenerating the client, you will need to make a number of changes in the code in order for the new client to work.

Defining WS-Stack

To migrate to a remote WS-Stack

1. Copy the applinx-wsstack.jar file (can be found in the ApplinX installation wsstack\WEB-INF\lib directory) and the applinx-shared.jar file (can be found in the ApplinX installation shared\lib directory) from the ApplinX installation, to the WS-Stack Web application, WEB-INF\lib directory.
2. In the Server Properties dialog box, configure the WS-Stack remote parameters: enter the host name, the Tomcat port where WS-Stack Web application is deployed, the URL of the servlet that the WS-Stack uses for deployment tasks and the user name and password used by WS-Stack.

Procedure Clients

Procedure client is a code representation of an ApplinX Procedure Group which can be integrated into any Java/.NET development environment to execute ApplinX procedures.

ApplinX server can generate three types of procedure clients: C#, VB.NET and Java clients. The generated client contains four types of classes:

- Service class (per procedure group): Contains methods for each procedure under the relevant procedure group. Calling a certain method will execute the corresponding ApplinX procedure.
- Request class (per procedure): Encapsulates all the inputs of that procedure.
- Response class (per procedure): Encapsulates all the outputs of that procedure.
- Data Structure class (per Data Structure): Using a Data Structure in a procedure generates a separate class. This class contains as field members all the attributes of that Data Structure.

This section covers the following topics:

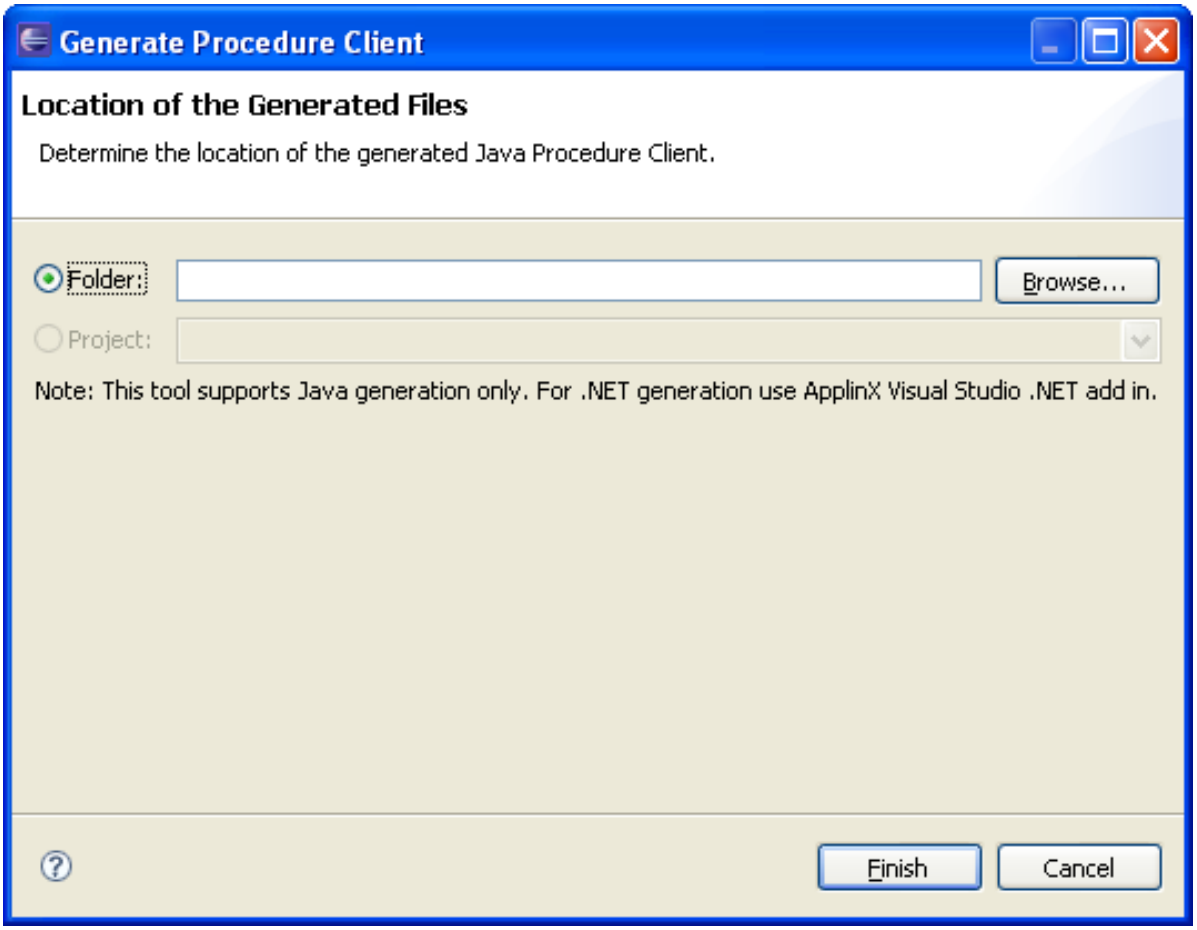
- Creating a Secure SSL Connection between a Procedure Client and ApplinX Server

Generating a Procedure Client

Generating a procedure client can be performed either using the Eclipse Designer (JSP) or the .NET add-in.

To generate a procedure client (JSP)

1 In the Root directory of the relevant application, right-click on a procedure group and select **Generate Procedure Client...**. The *Generate Procedure Client* wizard is displayed.



2 Browse and locate the directory where the generated code will be saved.

3 Click **Finish**.

The Procedure Client Structure

Java Client Structure

For Java clients, each class (as discussed above) is generated as a separate Java file. Therefore, when using Java, the namespaces defined in ApplinX are used as package names for each class. If no namespaces are defined, in ApplinX the package name will be the same as the procedure group's name.

For example: If procedure group "Test" is defined in ApplinX under the namespace TestNamespace, then the generated service class will be named TestService and its package name will be Test-Namespace.

Code example:

```
Package TestNamespace
Public Class TestService
{
Public method A
Public method B
}
```

.NET Client Structure

For .NET clients (C#) a single class file is generated, which contains all the classes aforementioned. In the case where no namespaces entities were defined in ApplinX, all the classes will be nested inside a single namespace, whose name will be the same as the Procedure group's name. However, if a procedure group or Data Structure were defined under a specific namespace in ApplinX, they will be nested inside that namespace in the generated class file.

For example: If the procedure group "Test" is defined in ApplinX under the namespace `TestNamespace`, then the generated service class will be named `TestService` and will be nested under `TestNamespace`.

Code example:

```
Namespace TestNamespace
{
Public class TestService
{
Public method A
Public method B
 }
}
Namespace BusinessEntityNamespace
{
Public class MyBusinessEntity
 {
Public field A
```

```
Public field B
 }
}
```

Developing with the Procedure Client API

This section explains how to use the procedure client classes in your application. This example illustrates how to run a simple flow procedure that gets a string array as input and returns a Data Structure with two attributes as output. Refer to the SOA demo application to see an implementation.

Note: The code in the example below is written in C# syntax.

1. Create a new instance of the Request class.

   ```
   ProcedureRequest request = new ProcedureRequest ();
   ```

2. Create a new instance of the Data Structure class.

   ```
   BusinessEntityClassName InputObject = new BusinessEntityClassName();
   ```

3. Set the attributes of the Data Structure with values

   ```
   InputObject.Attribute1 = Value1;

   InputObject.Attribute2 = Value2;
   ```

4. Set the attribute of the request object (created in step 1) with the Data Structure object (created in step 2 and 3).

   ```
   request.InputAttribute = InputObject;
   ```

5. Create a new instance of the service class.

   ```
   ServiceClass service = new ServiceClass();
   ```

6. Call the method that runs the relevant procedure. Pass the request object to that method. The method will return a response object.

   ```
   ResponseClass Response = service.Procedure(request)
   ```

7. The string array output can be extracted from the response object and used anywhere in your application.

   ```
   ForEach (string str in Response.Output)

   {

   Console.Out.Writeln(str);
   ```

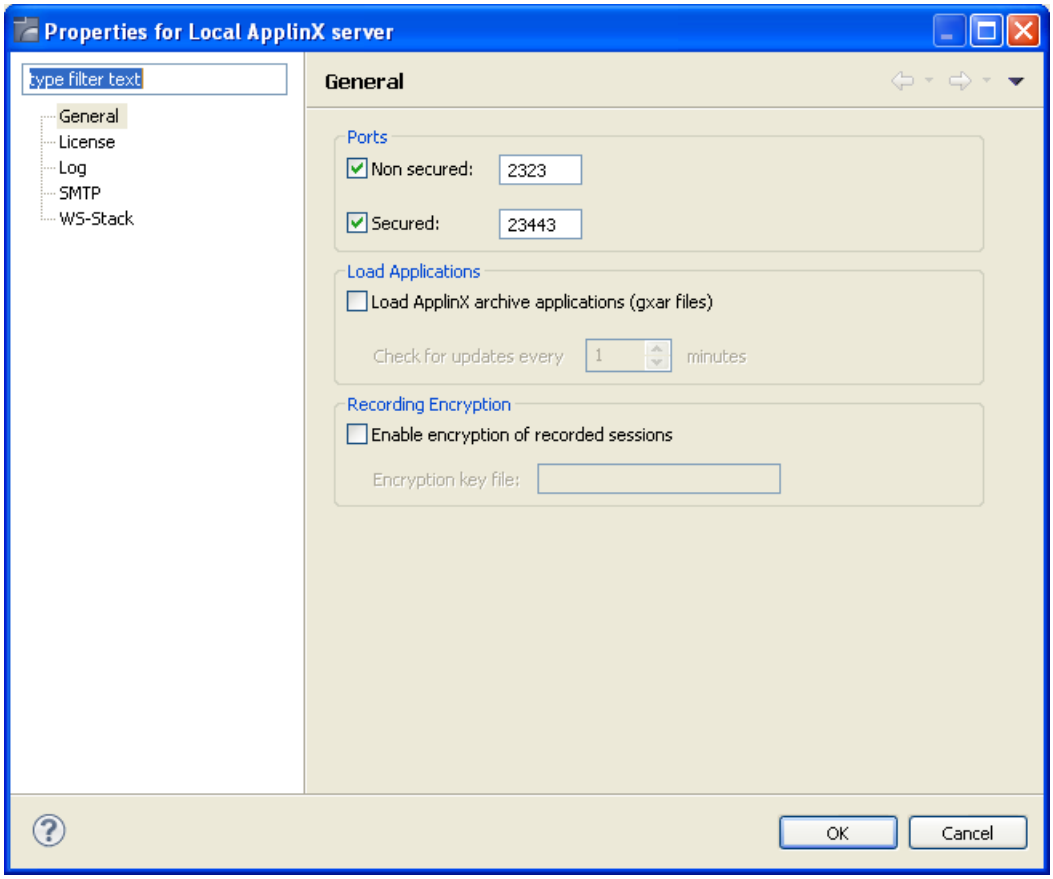
Creating a Secure SSL Connection between a Procedure Client and ApplinX Server

ApplinX enables a secure SSL connection between a Procedure Client and ApplinX server. To enable this feature you will need to first configure the server to support working with a secure SSL connection and then make the necessary changes in the code that executes the service.

To configure the secure SSL connection:

1 Configure the server:

1. In the Server Properties (right-click on the server), General tab, select the Secured port check box and enter the port number. By default the port number is 23443.



2. Restart the ApplinX server.
3. Check that you can successfully connect using a secure connection; login to the ApplinX Administrator and select the Secured connection checkbox.

2 After generating a Procedure Client, access the code that executes the service (when using ApplinX Web framework it is located in the "code behind" of the page (*<page_name>.java*/*<page_name>.cs*)) and add the following two lines beneath your service:

```
service.setApplinXServerPort(<secured port number>);

service.setSecured(true);
```

<secured port number> being the port number defined in Secured port configured in step 1.a.

18 Connection Pools

Connection pools enable you to immediately get a host connection that is ready in a specific screen, the "initial screen". This saves the time of establishing the connection with the host and navigating to the relevant screen. See Navigation. This chapter applies to Web enablement and SOA scenarios running a path procedure; it covers the following topics:

See also: Connection Pool and Connection Information Sets in the *Reference Guide* | **Connection Information Sets** in this manual.

Creating a New Connection Pool

To create a new Connection Pool

1 Select the relevant application.

2 In the **File** menu select **New>Entity>Connection Pool**. The *New Connection Pool wizard* is displayed.

3 Enter a name for the screen, a suitable description and determine the folder where the screen is to be located. Click **Finish**. You have now created a new Connection Pool in the repository.

4 Edit the settings in the Editor area as detailed in the Connection Pool reference.

Changing the Initialization Mode of a Connection Pool

To change the Initialization Mode of a Connection Pool

1 Double-click on the relevant Connection Pool.

2 Select the **General** tab.

3 In the **Initialization mode** field, select the relevant mode:

- Manual: An administrator manually initializes the connection pool (the connection is initiated by selecting the connection pool in the Management node, and then choosing Start Pool from the right-click shortcut menu).
- When first accessed: When Web applications or sessions request to connect to this connection pool.
- Automatic: Automatically initializes the connection pool when the application is loaded.

Starting and Stopping Connection Pools in the Designer

You can manage connection pools from the ApplinX Explorer of the Software AG Designer.

To start a connection pool

- Select the connection pool, and from the context menu choose **Start Connection Pool**.

To stop a running connection pool

- Select the connection pool, and from the context menu choose **Stop Connection Pool**.

Changing the Log Level

Fine-tuning connection pool parameters or identifying problems are reasons you may want to change the level of detail that the log displays. It is possible to set a different detail of logging for each connection pool: None, Errors, Warnings, Information and Details. The server logger should be set to no less that the **Normal** log level, in order to see connection pool logs. Once the project is in the production phase, **Error** level is the recommended level to use.

Defining an Initialization Path

An initialization path is a path that can be executed on any new connection to the host that navigates the connection to one of the Initial Screens. In order to select a path from different folders, click the folder selection button.

Open the Navigation tab of the relevant Connection Pool and select the path/path procedure from the Initialization drop-down list.

Defining a Termination Path

A termination path is a path that can be executed on any connection to the host (used or new) that should be performed before destroying a connection (for example, a host-side logout procedure). A termination path is triggered automatically by the ApplinX Server in the following situations:

- when a connection is canceled
- when a connection is disconnected, with "disconnect after usage" configured

- when the connection pool is stopped or exceeds the number of allowed connections based on the pool policy and needs to reduce the number of available connections
- when the connection is trying to return to the pool but can't reach the initial screen; the termination path will be used to cancel the connection gracefully

To define a termination path, open the **Navigation** tab of the relevant connection pool and select the path/path procedure from the **Termination** drop-down list.

Connection Pool Lifecycle

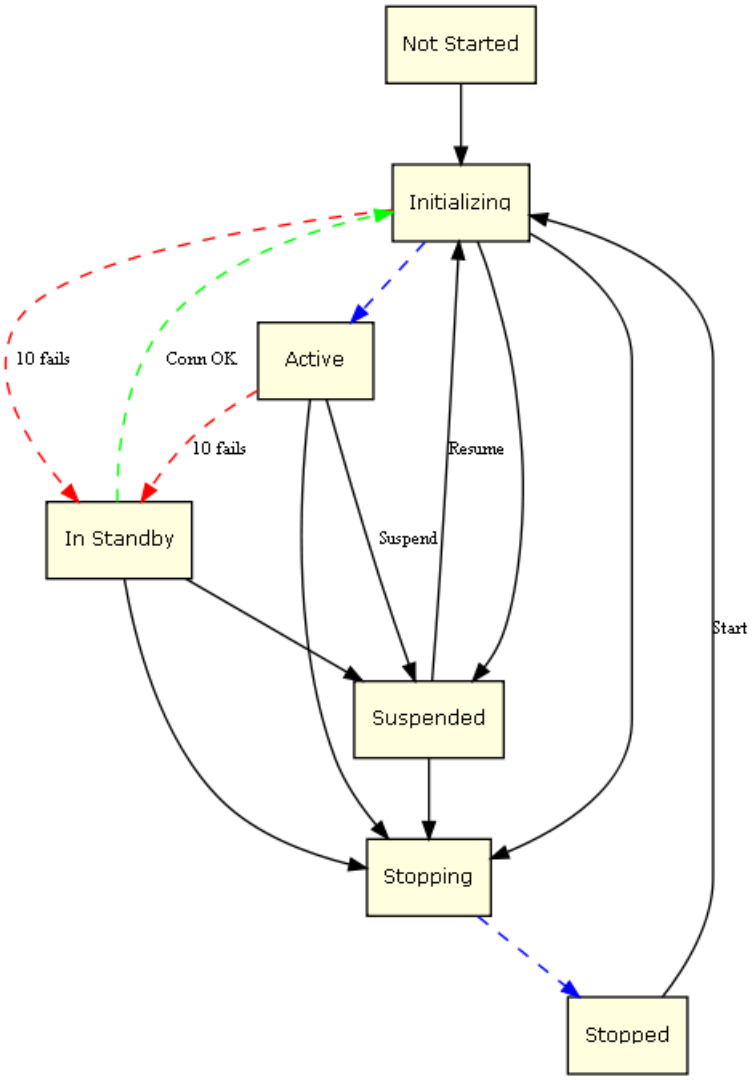
Connections in the pool automatically go through different states, according to the pool policy as defined in the Connection Pool Entity. The connection policy is built from:

- Pool size type: fixed, limited, flexible
- Keep-alive path and interval
- Run recycle anyway flag
- Disconnect after usage flag
- delays and timeouts

Other factors may also impact pool behavior.
See Pool Size Control Policy Considerations | License Key Concept | Connection Information Sets.

Connection Pool States

- The Connection Pool itself can have seven states:

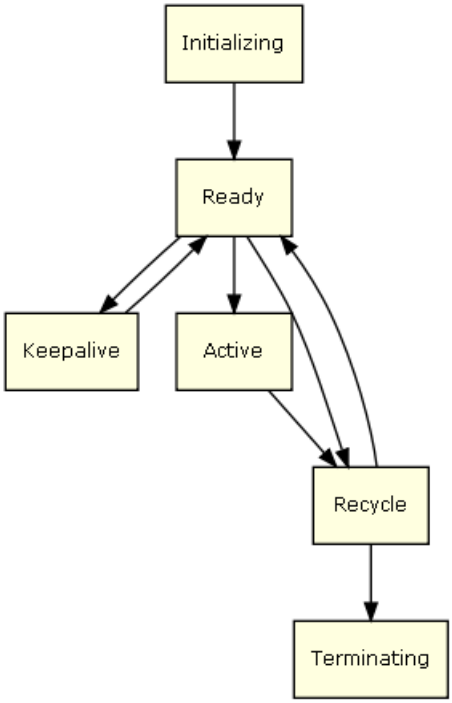


State	Description
Not Started	The connection pool was not initialized yet. If you request a connection, the connection pool will return an immediate error.
Initializing	The connection pool is trying to reach Active status, but does not have any "ready" connections yet. If you request a connection, the connection pool will return an immediate error.
Active	The connection pool is working and managed to create at least one connection to the host.
In Standby	The connection pool had several consecutive errors trying to create new connections to the host. The connection pool will continue to try to connect if user requests arrive, but will not initiate new connections otherwise. If a new connection is successfully created, the status will automatically change to Active.
Suspended	The connection pool is blocked for new users and does not maintain its connections. If you request a connection, the connection pool will return an immediate error.

State	Description
Stopping	The connection pool is trying to reach Stopped status, but still has connections in different stages of termination. When all connections are down, the status will automatically change to Stopped. If you request a connection, the connection pool will return an immediate error.
Stopped	The connection pool does not have connections or maintenance. If you request a connection, the connection pool will return an immediate error.

Note: the Stopped state will only happen after Stopping; the Initializing state will happen before Active. These are "in between" states, typically not lasting more than a couple of seconds.

- Host connections inside the connection pool can have six states:



State	Description
Initializing	Host connection will be in initializing state when created and until it reaches its initial screen, usually while running the initialization path. If you are using a connection info set: the parameters of a single record are retrieved.
Ready	Connection connected to the host but not yet serving any requests (session/user). Waiting on the initial screen for a request.
Active	Attached to a session/user, currently serving a request.
Keep-alive	If a keep-alive path is defined in the connection pool, the connection will be in this state while the keep-alive path is running (typically no longer that a few seconds).

State	Description
Recycle	If a recycle path is defined in the connection pool, the connection will be in this state while the recycle path is running.
Terminating	Disconnecting the connection. If a termination path is defined in the connection pool, it will be executed. If using a connection info set: this will release the parameters of the record used.

Note: Connections can also appear to be in Broken state. This is not a "real" state, it just indicates that the connection cannot get to a Ready state. Broken connections do not count with regards to license capacity and do not use any system resources. Broken connections will be automatically cleared after a certain time. See **Connection Pool Troubleshooting** for more information.

Connection Pool Troubleshooting

Symptom	Explanation	Possible Reasons	How to Check
All my connections are broken	Host sessions become *broken* if they cannot get to Ready state. Ready means that: 1. The connection is alive. 2. If an "initial screen" is defined by the pool, the session must get there.	Host is unavailable.	Try to connect to the host without the pool.
		Initialization path needs to be modified.	Create a connection outside the pool, run the initialization path and verify it gets to the "initial screen".
Connections disconnect after use but I want to reuse them	The pool may be configured to allow reusing a host connection that was used by a previous user. This would require returning the connection to the initial screen, either by implementing a robust navigation logic inside the invoked procedure or in the recycle path.	The **Disconnect after usage** option is checked.	Observe the state of the Disconnect after usage checkbox in pool tab of the connection pool.
		The logic of the user activity or the recycle path causes a disconnection from the host.	Capture a trace file for the user activity and recycle path. Observe that the host session is not terminated during the invocation of user activity or the recycle path.
		When a connection is returned to the pool and it is not in the "initial screen", and there is no recycle path that successfully navigates to the initial screen.	Capture a trace file for the user activity and recycle path. Observe that at the end of the user activity and the recycle path invocation, the connection is in the initial screen.

Symptom	Explanation	Possible Reasons	How to Check
Connections remain Active after use	Upon finishing the user activity on a pool connection, the state of the connection would change according to the pool configuration and the state of the user session. The state of the connection would remain active if the session used by the pool has not ended.	A flow procedure creates an emulation session on a pool connection, but the "end session" node is not called on that session.	Verify the existence of an "end session" node in the flow procedure that creates the emulation session. Verify that the "end session" node is reached by capturing a trace file or by logging the invocation of the flow procedure.
Connections remain Terminating after use	A pool may be terminating a connection based on the pool's configuration, connection count and the status of the session. When doing so, the connection status would become "Terminating" and the termination path would run on the connection. After the termination path is completed, the connection is removed from the pool.	An exception in the termination path is causing the host connection to either get stuck or to terminate before completing the path.	Capture a trace file that includes the invocation of the termination path and observe that the path has completed successfully. Search for exceptions in the server logs that occurred during the invocation of the termination path.
A used connection is disconnecting while being used		The host is disconnecting the pool connection based on the user activity or due to the host state.	Capture a trace file for the user activity, observe the last packet transmissions between the server and the host. Look in the server log for an error indicating a socket close around this timestamp. Also ask the host administrator to inspect the host's log for disconnections.
		Another host connection is using the same host credentials/device name and hijacking the session.	When capturing trace files, you would notice a trace file created at the time when a previous trace file is closed. Both traces will include send sections containing the same credentials or device name.

Note: When recording trace files to capture the symptoms mentioned above, we recommend using the following variables in the trace file name: connection ID (`%c`), session ID (`%u`), creation time of file (`%t`). See *Recording Trace Files* for more information.

19 Connection Information Sets

The connections in a connection pool are created before an end user actually connects to ApplinX. Therefore, connection or login parameters that are usually supplied by the user (such as user ID, user password, device name, etc.) must be supplied by a different source. This source is the Connection Info Set (also known as Connection initialization parameters), which supplies a pool of possible connection parameters required for the initial connection to the host (such as the device type or host address) and for the execution of the connection pool initialization path when it exists (such as the required user name and password).

Each information set may list a number of records. When initializing a connection, the parameters of a single record are retrieved. The number of times that a single record can be used concurrently is defined in the Repeat column of the Connection Information Set dialog box. When set to zero, this record can be used an unlimited number of times. More than one connection pool can use the same connection information set for its connections, though the total number of connections using the connection information set will not exceed the repeat value.

To define a Connection Information Set

1 Select the relevant application.

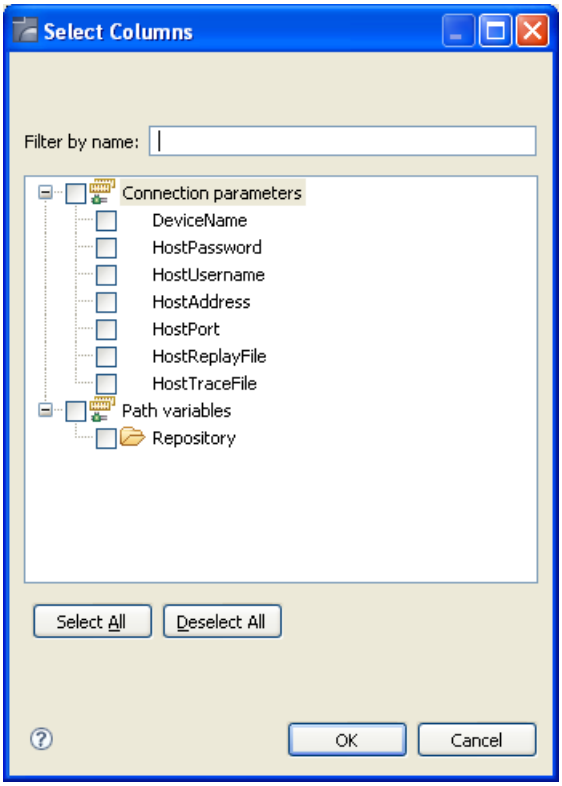
2 In the **File** menu select **New>Entity>Connection Information Set**. The *New Connection Information Set wizard* is displayed.

3 Enter a name for the Connection Information Set, a suitable description and determine the folder where the Connection Information Set is to be located. Click **Finish**. You have now created a new Connection Information Set in the repository and it is displayed in the Editor area.

4 Click **Add Record** to add additional rows to the table. Each row represents a connection information set record. Click on a row to edit the parameters of the row.

5 Determine the Repeat limit: In the Repeat column determine the number of times that a single record can be used concurrently. When set to zero, this record can be used an unlimited number of times.

6 Click **Define Columns** to display the Select Columns dialog box.



Select and/or remove your selection to determine the columns to be included in this Connection Information Set.

Select Connection parameters to define parameters required for initializing a session (e.g. device name, host name etc.). Select Application fields and variables to define parameters that are required by the initialization path used in connection pools.

Setting a password column

When a column is set as a password column the values in this column are scrambled in the database and appear as asterisks (*****) instead of the actual typed characters. In addition, the password values are not displayed in the log. Any of the columns can be set as a password column, except the first two columns (ID and Repeat) and Application Parameter columns. Once you set a column as a password column, you cannot change the column back again to be a non-password column. In order to remove the password feature you must delete the column and add it again with its entire cell content.

1 Access the relevant Connection Information Set.

2 Click on Set Password.

3 In the pop-up window displayed, select the relevant column. This option may not be available for some columns: Application parameters, the first 2 columns (ID and Repeat) and columns that are already password columns.

Note: The connection information set parameters can also be changed using the API detailed in the Administrative Web Services API

20 Data Structure

A Data Structure is an object that is characterized by attributes. Once defined, the Data Structure entity can be used as an input or output variable in Flow Procedures or as an attribute of a Data Structure.

To create a new data structure:

1 Select the relevant application.

2 In the **File** menu select **New>Entity>Data Structure**. The *New Data Structure wizard* is displayed.

3 Enter a name for the Data Structure, a suitable description and determine the folder where the Data Structure is to be located. Click **Finish**. You have now created a new Data Structure in the repository and it is displayed in the Editor area.

4 Click Add Structure to add a structure. Enter a name, description, select a type and determine whether it is an array.

5 Click Add Attribute to add an attribute. Enter a name, description, select a type, determine whether it is an array and provide a default value.

6 Use the Move Up and Move Down links to change the order of the data structures.

21 Session Data

Session Data is an entity used to save session context information. Each application has a single Session Data entity, located in the Repository folder. New applications are created with an empty Session Data entity which cannot be deleted. The Session Data entity enables defining variables and their types in the session context during design time. In Runtime, a Session Data object is created for each session, and it is possible to set and then use the variables' values in the current session. When using these variables in Connection Pools, whenever a connection is returned to the pool, the Session Data is reset with the default values.

Note: When executing a path using the Execute Procedure node, by default a new session is created for the executed procedure, therefore a new Session Data object is created for the executed procedure. To use the same Session Data object as the Procedure, map the session ID to the executed procedure ID.

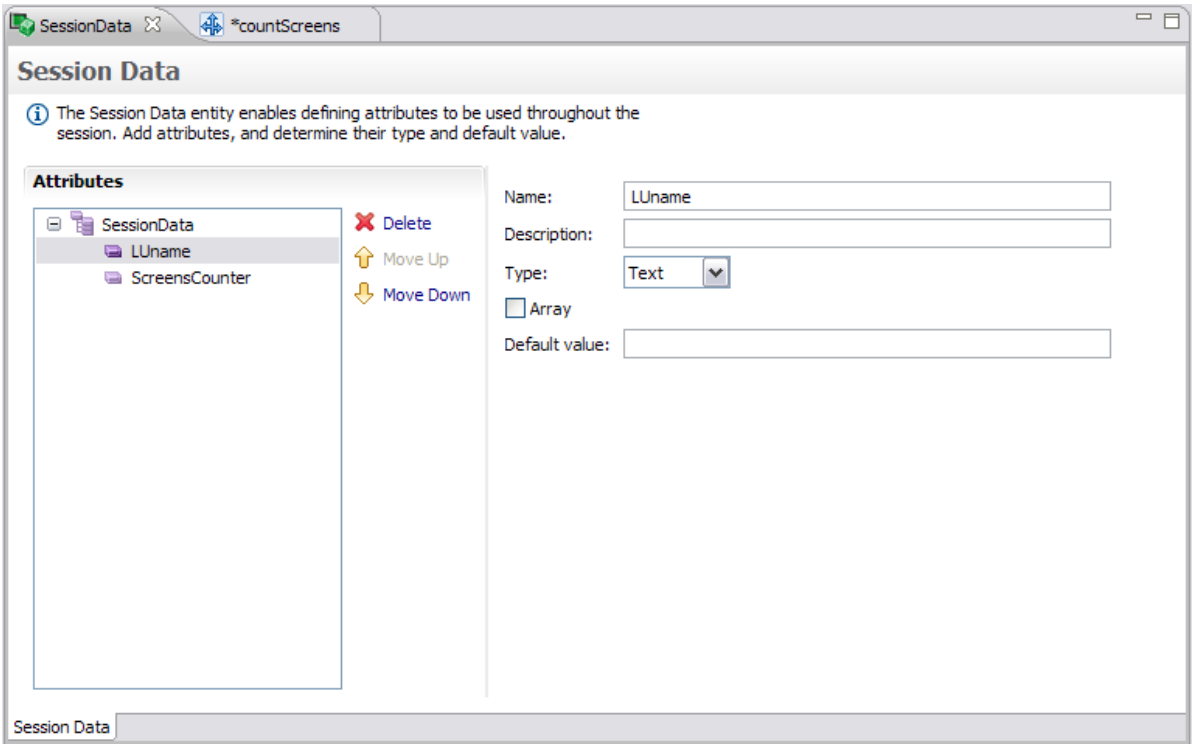
The Session Data entity can be used in the following entities:

- Path Procedure mappers
- Business Activity Mappers

For applications created in earlier ApplinX versions, when synchronizing the application database, an empty Session Data entity is created. The old database cannot contain a data structure named "SessionData" (and when such a data structure exists, it must be renamed prior to the import process).

To define variables and types of variables:

1 Within the Repository node of the application, double-click on SessionData. The Session Data entity is displayed.



2 Click **Add Attribute**. An attribute is added to the Session Data.

3 Provide a name for the attribute and a relevant description.

4 Select the type of attribute: Text, Long, Boolean, Double, Integer, Float, Byte or Date.

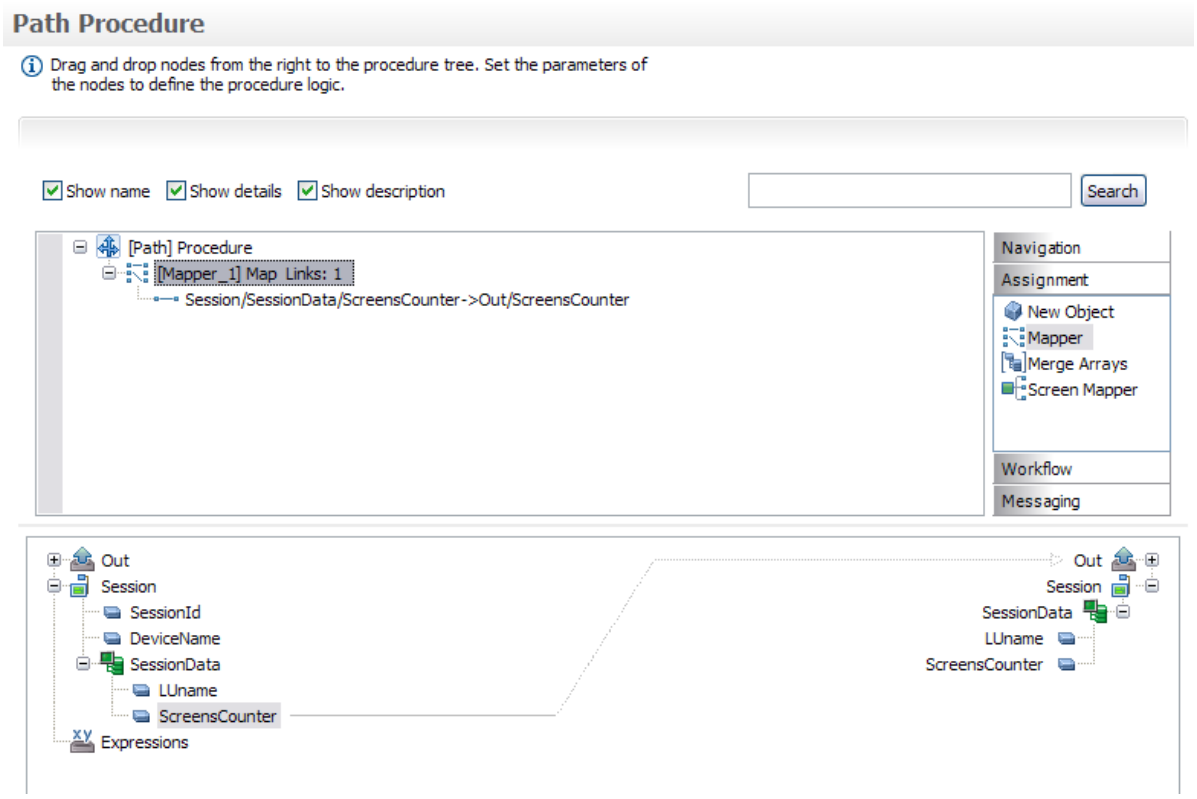
5 Determine if it is an array.

6 Enter a default value.

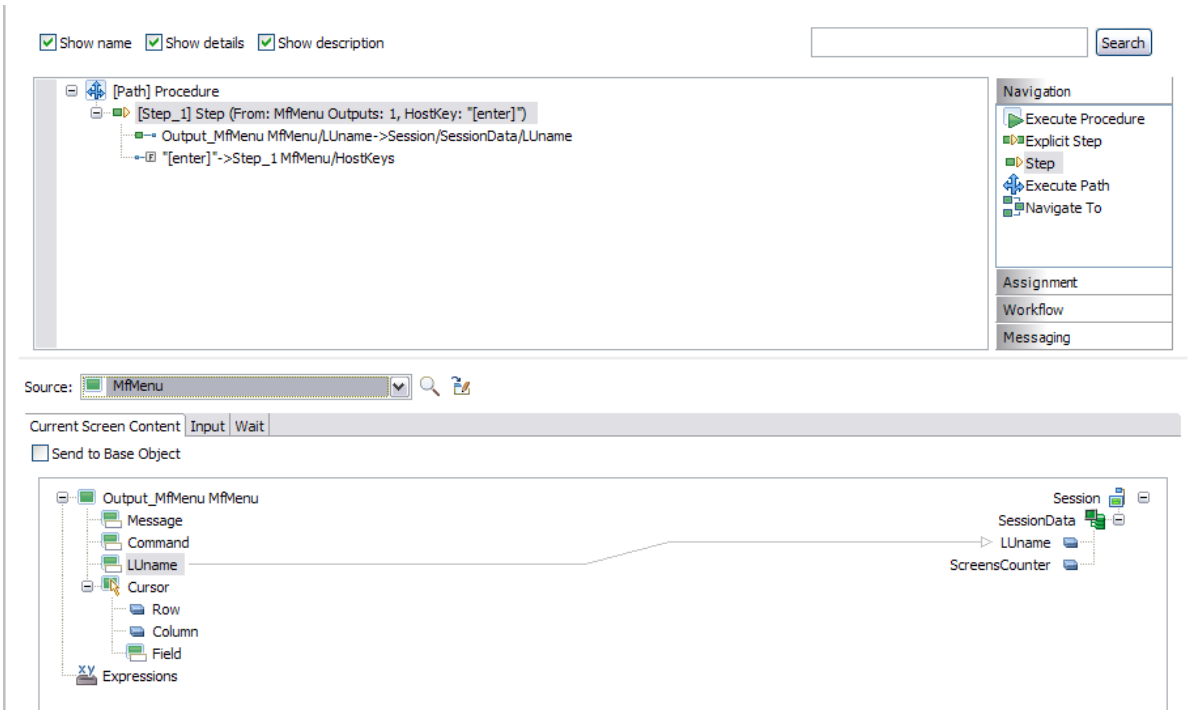
To Set and Use the Variables:

- Setting and using the values is implemented by mapping to or from the session data variables (refer to the Mapper).

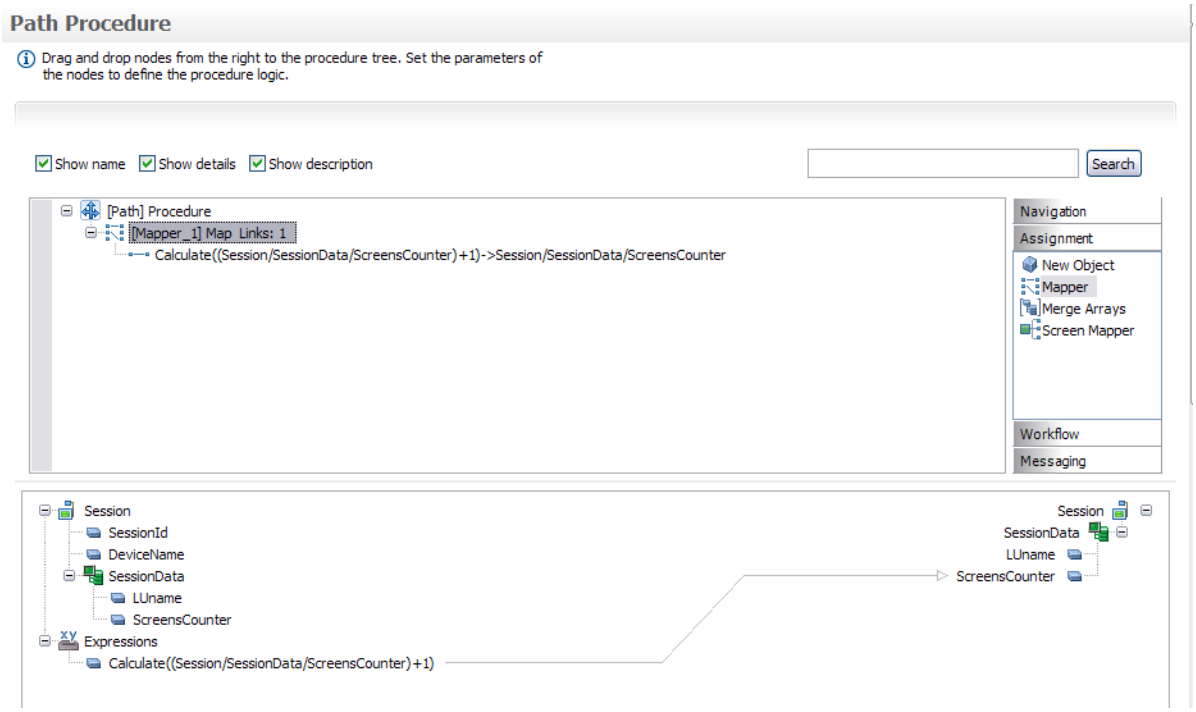
  The following illustrates mapping from the Session Data to the Procedure's outputs.



The following illustrates mapping from the screen's field content to the Session Data



The following illustrates mapping from an Expression which uses the Session Data, to the Session Data.



22 Database Connection

Database entities are used for direct database access, in flow procedures. Once the database connection is defined, it is possible to connect to it in any flow procedure and perform execute, select, commit and rollback statements.

To create a Database Connection

1 Create a new Database Connection via the **File>New>Entity** menu item.

2 Enter a name and description and click **Next**.

3 Select the type of database that will be used as the application's database connection.

Database	Driver	Requires Driver Files
Apache Derby	org.apache.derby.jdbc.EmbeddedDriver	
SQL Server	com.Microsoft.jdbc.sqlserver.SQLServerDriver	Yes
Oracle	oracle.jdbc.driver.OracleDriver	Yes
MySQL	org.git.mm.mysql.Driver	Yes
DB2	COM.ibm.db2.jdbc.app.DB2Driver	Yes

4 Click **Finish**. The Database Connection details are displayed.

23 Web Application Manager

Following are the topics related to the Web Application Manager:

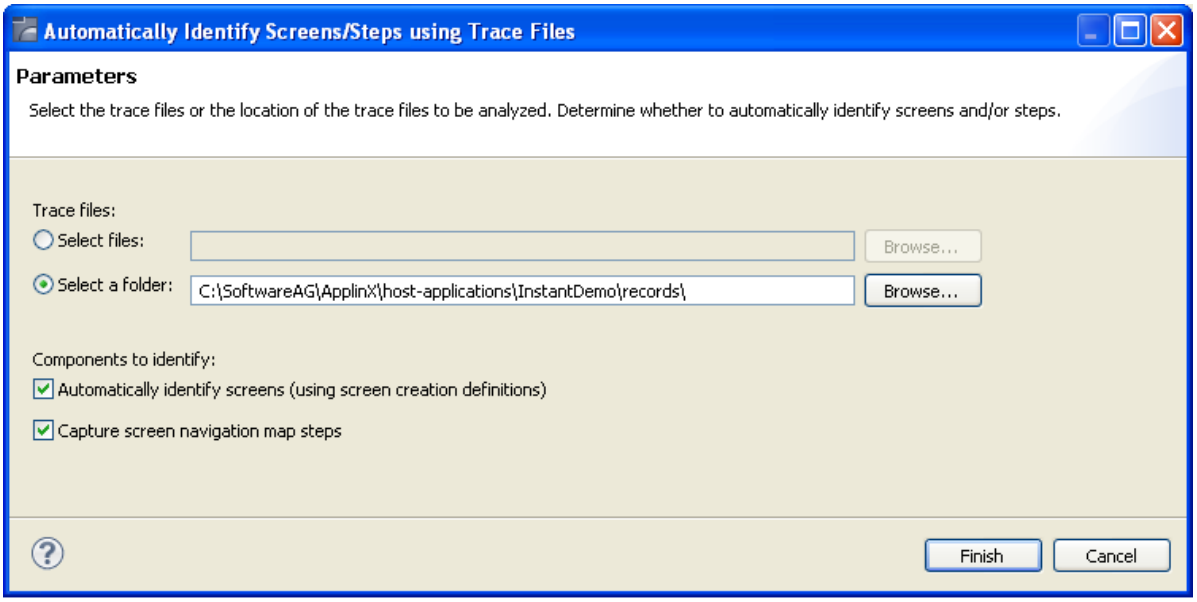
- Creating an ApplinX Web Application using JDK and Tomcat (no IDE)
- Creating an ApplinX Web Application using .NET SDK 2.0 (no IDE)
- Upgrading an Existing JSP Web Application
- Upgrading an Existing .NET Web Application
- Installing the Demo Web Application using Tomcat

24 Automatically Identifying Screens and Steps using GCT Files

Using trace (GCT) files, you can automatically identify screens. In order to do this, you must ensure that a Screen Group includes a **Screen Creation Definition**.

To automatically identify screens using trace files:

1 Right-click on the relevant application, and select **Automatically Identify Screens/Steps using Trace Files...**. The wizard is displayed.



2 Select whether to analyze files or a folder. Locate the files/folder.

3 To automatically identify screens, select the **Automatically identify screens** check box.

4 To automatically identify steps, select the **Capture screen navigation map steps** check box.

5 Click **Finish**.

The Console displays the outcome of the process.

Note: When identifying a new screen, the screen name is taken from the area you defined in the Screen Creation Definitions. If a screen with this name already exits, the newly identified screen's name will be the name as of the existing screen followed by an incremental number. If the area you defined in the Screen Creation Definitions is empty, the screen will be named: "Screen" and then an incremental number.

Refer to **Screens** and **Application Maps** for in-depth documentation on each of these subjects.

IV

25 Working with JSON Files

See also *Managing Test Data*.

Introduction

JSON is a text file that can convert to any JavaScript object and can be sent to a server. You can also convert any JSON received from the server into JavaScript objects. This way you work with the data as JavaScript objects, with no complicated parsing and translations. In ApplinX, you could generate the procedure with initial file *Data.json* file containing the default inputs and assertions given by the procedure. For example, the following input and output parameters:

```
{
"inputs":
   {
   "Code":"cu"
   "Password":"demoy2"
   "UserId":"demo02"
   "structure1":
      {
      "Birthday":
      "Customer_ID":
      "First_Name":
      "Last_Name":
      "Sex":
      "TotalPoliciesAmount":
      "Type":
      }
   }
"Assertions":
    {
    "Firstname":"Charles</Firstname>
    "House_Number":"12</House_Number>
    "Lastname":"Chappell</Lastname>
    "Mobile":
    "Municipal_Code":
    "Nationality_Long":"British"
    "Nationality_Short":"UK"
    "Occupation_Long":"Actor"
    "Occupation_Short":"81177"
    "PO_Box":
    "Person_Sex>M</Person_Sex>
    }
} ↩
```

Working with Arrays

You can also provide input using an array. The `runTest` method is provided for sending inputs from a given JSON file to the Base Object and verifing the assertions from the JSON file, using the response data.

Each input entry in the JSON file is converted to `GXElement` so that the procedure can run. Each output attribute given by the Base Object response is converted to a JSON object and compared with the assertion loaded from the resources folder automatically.

JSON Configuration File

The JSON configuration file enables you to change the following values for excuting the tests:

```
{
   "serverPort": "2323",
   "serverAddress": "localhost",
   "apxUser": "administrator",
   "apxPassword": "",
   "hostApplication": "SAG_MainFrame"
}
```

Parameter `hostApplication` is mandatory; all others are optional. If a parameter is not defined, the test will try to run with the default value.

If parameter `hostApplication` is not defined, the test will run on actual application host.

Working with Multiple JSON Files

The following mechanisms are provided to enable you to work with multiple JSON files:

* (asterisk)	any files
`f1.json`, `f2.json`...	list of files

The following rules apply:

- The file extention must be ".json".
- The list must not be recursive.
- The file name must be valid for the operating system.

Example:

```
public static Collection <Object[]> getTestData() {
    String[] fileNames = {"file1.json", "file2.jsOn"};
     return TestCase.getTestData(fileNames);
}
```

The default filter for test data is "*".

Comparing the Results of a Unit Test

If a failure occurs in a unit test, you can use a compare function to easily locate the source of failure.

To compare the actual results of unit test with the expected results

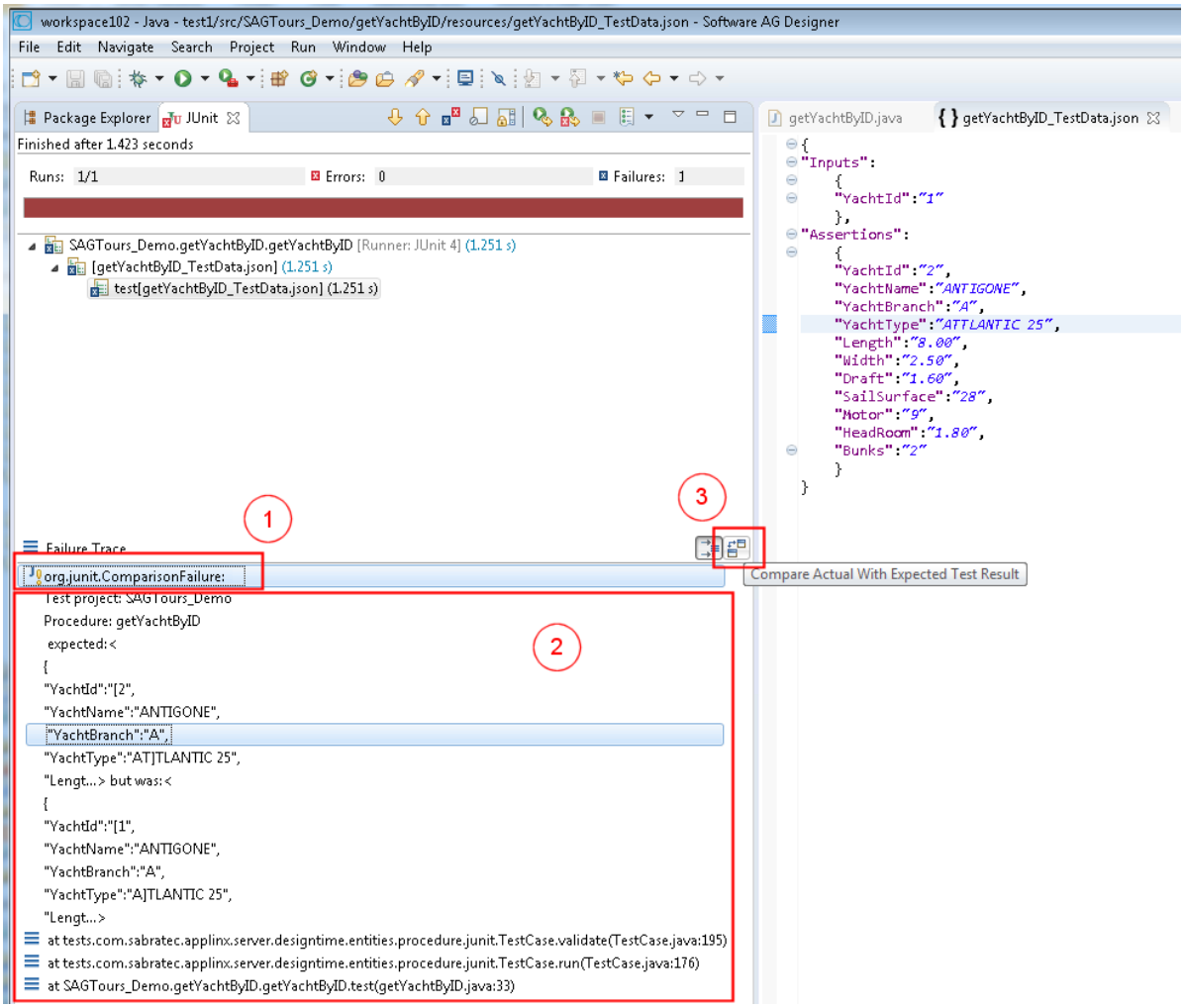
- Double click the line `org.junit.ComparisonFailure` in the JSON file. See (1) in screenshot below.

  Or:

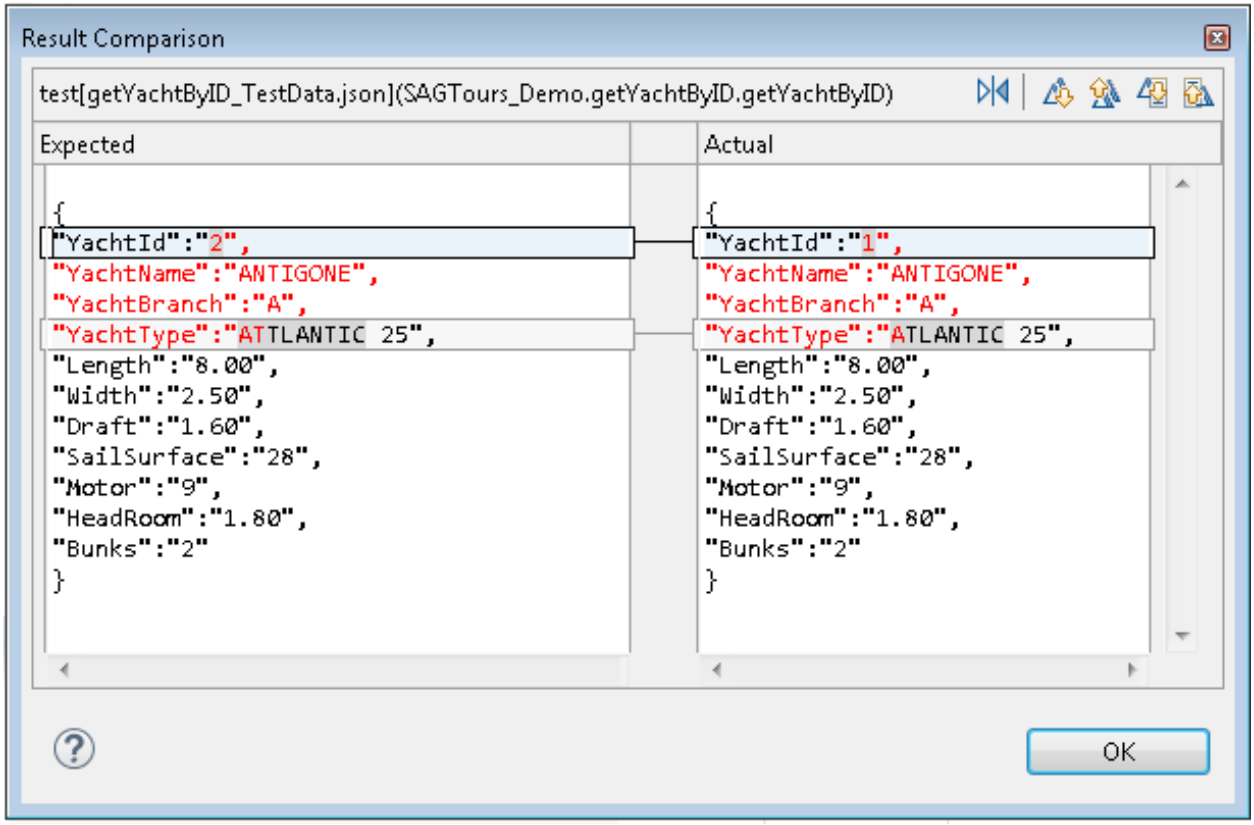
  Select the line in the marked rectangle area (2) in the screenshot below, and from the context menu choose **Compare Actual With Expected Test Result**.

  Or:

  Select the line and click the **Compare** icon  . See (3) in the screenshot below.



This produces the following output, where the discrepancies are highlighted.

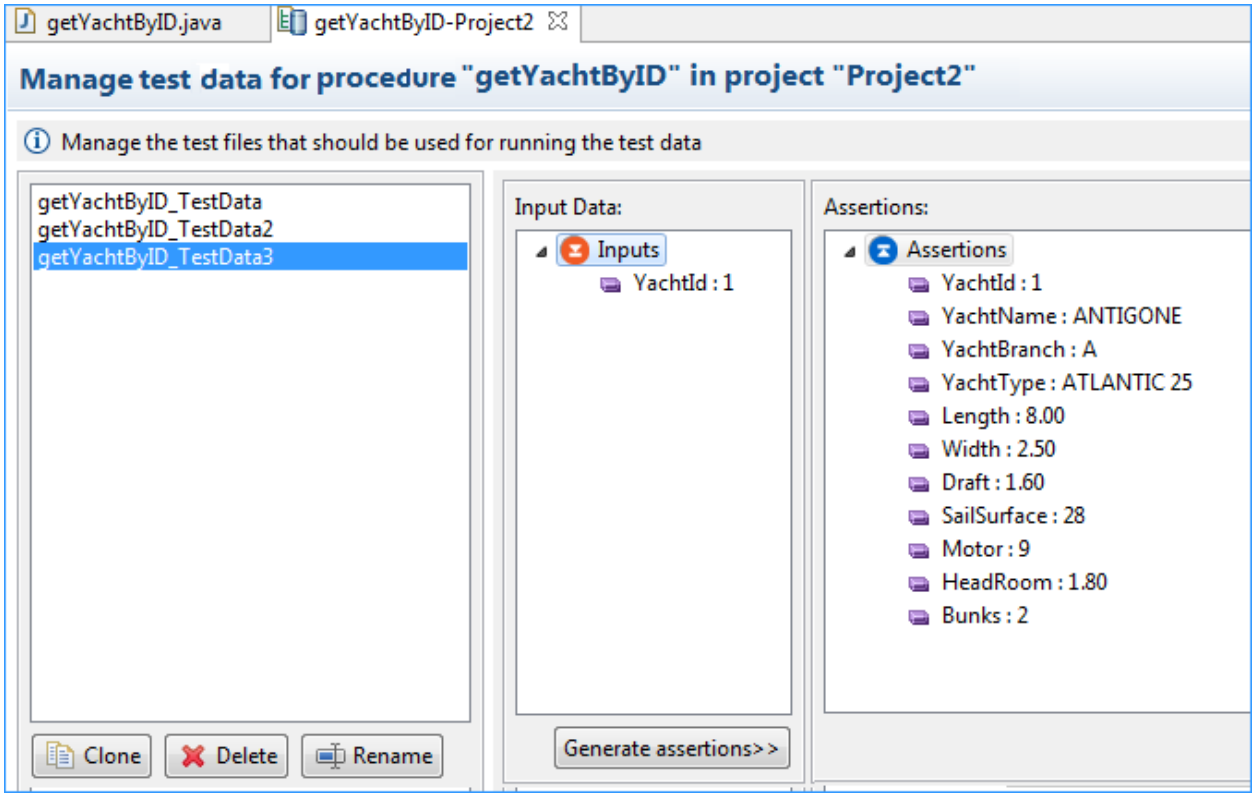

Result Comparison
test[getYachtByID_TestData.json](SAGTours_Demo.getYachtByID.getYachtByID)
Expected
Actual
{
"YachtId":"2",
"YachtName":"ANTIGONE",
"YachtBranch":"A",
"YachtType":"ATTLANTIC 25",
"Length":"8.00",
"Width":"2.50",
"Draft":"1.60",
"SailSurface":"28",
"Motor":"9",
"HeadRoom":"1.80",
"Bunks":"2"
}
{
"YachtId":"1",
"YachtName":"ANTIGONE",
"YachtBranch":"A",
"YachtType":"ATLANTIC 25",
"Length":"8.00",
"Width":"2.50",
"Draft":"1.60",
"SailSurface":"28",
"Motor":"9",
"HeadRoom":"1.80",
"Bunks":"2"
}
OK

26 Managing Test Data

Use the function **Manage Test Data Files** to create and manage test data files. An editor displays the JSON data of the generated test. With the **Clone** button you can create copies of the data and update the copies with new scenarios. When you save your changes, JSON files are updated/created and are in sync with the data of the editor. See also *Working with JSON Files*.

The editor window consists of three panes:

- **Left Pane**
  This pane lists the available data files. If you select a file, its contents will be shown in the main pane.
  - With the **Clone** button you create a copy of the selected data file. When this copy is saved, a new JSON file is created in the project directory.
  - With the **Delete** button you delete the data file.
  - With the **Rename** button you rename the data file.
- **Input Data Pane**
  Lists the inputs used in the selected test data file. The values are presented in an editable text field.
  - With the **Generate assertions** button you execute the procedure and fill in the assertions with the values received from the run.
- **Assertions Pane**
  All values to be asserted are listed here in the form of key + value. All values are in an editable text field, which you can use to update expected results manually and save the changes as new test data assertions.


getYachtByID.java
getYachtByID-Project2
Manage test data for procedure "getYachtByID" in project "Project2"
Manage the test files that should be used for running the test data
getYachtByID_TestData
getYachtByID_TestData2
getYachtByID_TestData3
Clone
Delete
Rename
Input Data:
Inputs
YachtId : 1
Generate assertions>>
Assertions:
Assertions
YachtId : 1
YachtName : ANTIGONE
YachtBranch : A
YachtType : ATLANTIC 25
Length : 8.00
Width : 2.50
Draft : 1.60
SailSurface : 28
Motor : 9
HeadRoom : 1.80
Bunks : 2