ApplinX User Guide

Reference Guide

Version 9.10

April 2016

WEBMETHODS

Document ID: APX-UG-REFGUIDE-910-20160402

Table of Contents

I Application & Host Parameters

1 ApplinX Eclipse Preferences

The ApplinX Eclipse preferences can be accessed via the **Windows>Preferences** menu in the Software AG node.

Remember passwords
Remembers the server passwords. When this check box is not selected the login pop-up is always displayed.

Always use the highest prioritized Screen Group (when creating Screens using Screen Creation Definitions)
When creating screens based on screen creation definitions which are defined within screen groups, you can determine to always use the highest prioritized screen group and not to ask the user every time which screen group to use. When this check box is not selected the dialog box requesting you to select the relevant screen group will always be displayed.

Switch to ApplinX perspective upon ApplinX wizards completion
When in an Eclipse perspective which is not the ApplinX perspective, and you run a wizard, once the wizard is completed, the perspective will change to be the ApplinX perspective.

Show Application Map on startup
Determines whether the APplication Map view is displayed when opening an application. By default the Application Map view is displayed.

ApplinX installation directory
Indicates the directory where ApplinX is installed.

JDK location
Indicates the directory where JDK is located.

Software AG tomcat location
Indicates the directory where Software AG Tomcat is located.

2 Application Configuration Parameters

The application properties dialog box can be accessed by right-clicking on the relevant application and selecting **Properties**. Here you edit the basic properties as well as advanced properties:

General Application Parameters

This node is the first node of the application properties and provides the most basic application information. Right-click on an application to access the general properties.

Application name
A unique name defined when first creating the application. The application name is read only here but can be changed by right-clicking on the application and selecting **Rename...**.

Description
A brief description of the application.

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).

Host Parameters

Host Parameters are described in detail in **Host Configuration Parameters**.

- General Host Parameters
- Offline
- Recording
- Send Options
- RPC
- Terminal Emulation Proxy

- VT Parameters

General Host Parameters

Host name
The name of the host that is to be connected to your application (obligatory field).

Address
The host's TCP/IP address (read only).

Port
The TCP/IP port to which the host is listening (read only).

Device Type
The device type of terminal that ApplinX emulates (read only).

Protocol
The protocol used to access the host (read only).

Model
The number of characters per column and per row, in the host's window (read only).

Code Page
The code page number of the required language for this application (read only).

Activity

Non-activity timeout
ApplinX can be defined to disconnect after a certain period of non-activity. When selecting unlimited, the session will stay connected even when there is no activity.

Wait condition timeout
The default amount of time, in milliseconds, for which the application waits to receive a screen from the host. For example, in paths, if the user defined a target screen, and it was not reached within this timeout, the user will get an exception. An additional example is in the frameworks, where the user will receive the last screen, and will continue waiting (as the host is still locked). Possible values are from 1000 to 999999 ms.

Flicker
An amount of time, defined in milliseconds, in which the application waits for the host to send information. This is necessary, as information may be sent in a few chunks of data, before declaring that it has received the entire screen. This is a value defined for all the application screens, and can be bypassed using the Wait Conditions. (By default, this value is set to 0). Possible values are from 0 to 10000 ms. Refer to Handling Flickering of Host Screens.

Blank screen timeout
The amount of time, defined in milliseconds, in which the application waits after receiving a blank screen, giving the host the opportunity to continue sending the next screen before returning to the client. This is very similar to the flicker parameter, the difference being that the flicker feature waits the complete amount of time defined as oppose to the blank screen timeout which finishes once the screen stops being blank. In addition, this setting only applies to blank

screens, and in this way, does not slow down the performance of the entire application. (Default value: 500 ms). Possible values are from 0 to 10000 ms.

Offline

Work offline
Indicates whether to simulate a communication with the host by using a pre-recorded file. By default, this feature is not enabled.

Files on server
Lists the trace files which are on the server. Click and browse to copy a file from your local machine to the server.

Simulate host delay
Allows 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). Available values: No delay, Simulate host, 500- 10000 ms (by default No delay is selected).

Recording

The Record Trace File feature enables creating a file, which will trace 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 if the data will be saved to a new file for every new connection or session. Identifying the separately saved files is possible by inserting identifying parameters in the file name (the session ID, connection time and/or connection ID).

Record display sessions (trace files)
Select this option to enable recording trace files.

Compress (create files in zip format)
Select this option to compress the file.

Encrypt (using server private key)
Select this option to encypt the file. In order to encrypt files, you must first define the encryption key (In the Server properties, General tab).

Suppress hidden fields
Conceals passwords and hidden fields in the trace file.

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

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 session ID, creation time and/or connection ID.

%u will insert the session ID.

%t will insert the time stamp of the connection.

%c will insert the connection ID.

Location of folder
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.

Open recordings folder
Opens the Windows Explorer and displays the location and list of existing trace files.

Note: The following section is not yet supported.

Limitations

Limit folder size
Select this check box to determine that the folder will be limited to a certain size and determine the maximum size of the folder.

Send notification when folder size reaches...
Select the check box so that mail will be sent once the folder reaches a certain percent of the maximum size.

Email
Enter the email address to which the notification should be sent.

Limit number of files to be stored in each sub folder
Select the check box to determine that the sub folders will be limited to contain a specific number of files and determine this number.

Create new folder when the sub folder reaches file limitation
Select the check box to create a new sub folder once the number of files in the current sub folder reaches the number determined in the previous field.

Send Options

The Send Options enable to configure how and whether a field is sent to the host when there are mismatches between the field and the host field.

Throw exception when field content does not match host field attributes
When the field does not match the host field attributes, for example, when the value to be entered is longer than the field length, or the field is not found, or the cursor position is not in an unprotected field, an error will occur.

Add trailing spaces to sent fields
ApplinX trims field content (removes padding spaces), so that the content is displayed properly in HTML input fields, and so that the user will receive only the meaningful content. However, some Mainframe hosts are sensitive to spaces and expect to receive from the client the spaces that the client received (unless they were overwritten by text entered by the user). To com-

pensate for ApplinX' trimming behavior and instruct it to send spaces to the host as needed, select this check box. This is relevant for 3270 only.

When field content is longer than host field length...
When the field content is longer than the host field it is possible to select the option that this field will not be sent to the host, or that the contents of the field will be cut to match the host field size.

RPC

Available in SOA applications only.

In the RPC tab, connection parameters for the ApplinX **Program Calls**. This is enabled only for AS/400 hosts.

Enable connection pool
Selecting this check box indicates that the server should maintain a pool of connections.

Number of connections in pool
The minimum and maximum number of connections it is possible to have in the pool at one given time.

Minimum available connections
The minimal number of connections in the pool that are ready to serve a transaction. If the number of available connections in the pool falls below this number, new connections will be created until the minimum is reached.

Connections created when pool is increased
The number of connections created each time the pool size is increased.

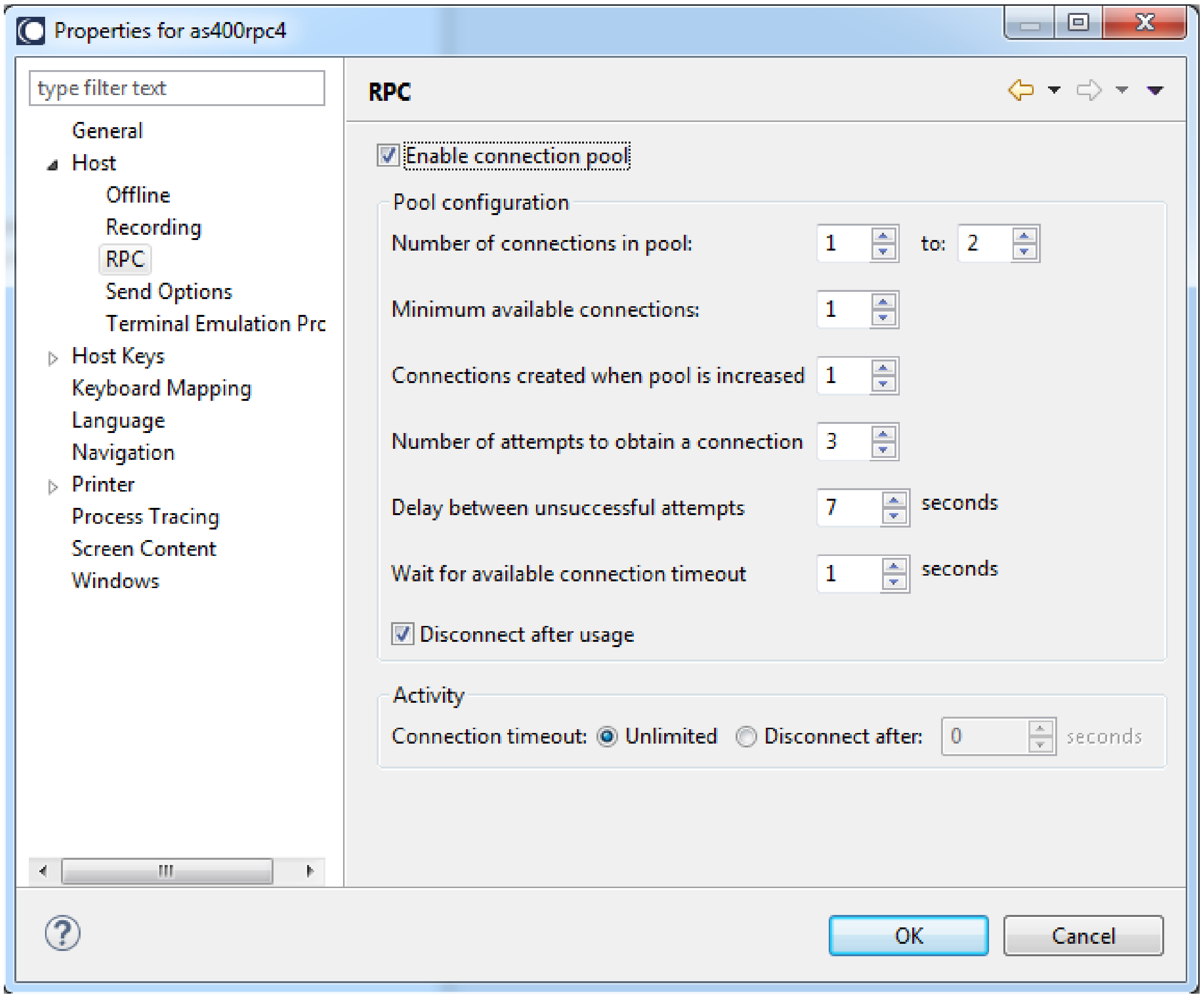
Number of attempts to obtain a connection
The number of attempts to create a connection.

Delay between unsuccessful attempts
The time, in milliseconds, before retrying to obtain a connection.

Wait for available connection timeout
If no connections are available in the connection pool, the length of time to wait for an available connection before a timeout message is sent.

Connection timeout
The total time, in minutes, that a connection can exist. Setting the value to "Unlimited" sets no timeout. After the set time elapses the connection is terminated.

Disconnect after usage
This checkbox determines whether to restart the connection once a user finishes using a connection or to terminate the connection when it is returned to the pool. When checked, the connection is terminated after usage.



RPC

[✓] Enable connection pool

Pool configuration

Number of connections in pool: 1 to: 2

Minimum available connections: 1

Connections created when pool is increased 1

Number of attempts to obtain a connection 3

Delay between unsuccessful attempts 7 seconds

Wait for available connection timeout 1 seconds

[✓] Disconnect after usage

Activity

Connection timeout: [✓ Unlimited] Disconnect after: 0 seconds

Terminal Emulation Proxy

This feature enables redirecting emulation network traffic to pass through the port configured in this screen (only relevant for AS/400 and Mainframe hosts).

Use Terminal Emulation Proxy
Select this check box to use the Terminal Emulation Proxy option.

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

Terminal Emulation Proxy port
Enter the port number through which you would like to redirect the emulation network traffic.

VT Parameters

Unprotected field navigation
Determine whether the arrow keys or any other key selected from the drop-down list manipulates the cursor when the unprotected fields are set (default: tab key).

Set cursor position
This is by default not selected. Determine whether to set the cursor position. If so, whether the arrow keys or any other key selected from the drop-down list sets the cursor position (default: tab key).

Use arrows to navigate to protected positions
Check this option when setting the cursor position with a key (not arrow keys). Allows you to set the cursor position in protected positions using arrow keys (default: checked when set cursor position is checked and key option is selected).

Use arrows to navigate within an unprotected field
Check this option when setting the cursor position with a key (not arrow keys), to set the cursor position inside unprotected positions using arrow keys. If this is checked, navigation to the first position of the unprotected field is achieved using the selected key. Continue navigation inside unprotected positions with arrow keys (default: tab key; checked when set cursor position is checked and key option is selected).

Automatically skip between unprotected fields
When checked, indicates the cursor skips automatically to the next unprotected field upon reaching the end of the current field (default) (happens when the field is completely filled with text).

Wait for unprotected field
When navigating to a certain field for sending its contents, the host sometimes moves the cursor to protected positions before finally placing them inside an unprotected field. When checked, ApplinX waits for the cursor to reach an unprotected field before continuing the navigation process. When unchecked, ApplinX continues the navigation process immediately after the cursor changes its position (even if the new position is protected).

Detailed log
When checked, a detailed description is logged to the ApplinX Server log file (default). It is highly recommended to check this during development time, however for production purposes it would be preferred to leave this check box unchecked to improve performance.

Send characters separately
Some VT hosts require input to be sent one character at a time. When checked, input characters are sent separately. When unchecked, input characters may be sent in one buffer.

Filler Char
Represents a character in the application that is treated like a blank. The default for new applications is underscore ("_"). The ApplinX server may use this character when, for example, erasing data from fields.

Host Keys

The Host Keys tab serves to define the host keys patterns (for example, PF/F/PA) that should be searched for in the application. To configure the host keys, click the Host Keys node in the Application dialog box. Click on the Manual node to customize host keys. Refer to Define Host Keys for additional information regarding to working with host keys.

Rows to search
Specify where to commence the search for PF keys. The number of patterns that can be defined for each application is unlimited. There are a number of predefined patterns which are available by clicking on the arrow next to the **Add Pattern** hyperlink.

Keys in successive rows only
Searches for host keys in successive rows only.

Search keys outside window
Searches for keys on entire screen-not only within the window frame.

Remove rows from screen
Removes the rows where the pattern was recognized.

Pattern table
Lists the defined patterns.

Pattern name
The name of the pattern.

Key prefix
The host keys' prefix (for example, PF/F/PA). It is assumed that one or two digits appear after the prefix. Maximum of two types of prefixes for each pattern is allowed.

Number of spaces between key and description separator
The number of spaces between the prefix and the internal separator. If a specific number is used, then only sets with that exact number of spaces will be included. Select "Any" to enable any number of spaces.

Key-description separator
The character or string separating the key and the description (for example "=").

Pattern separator
Separates between each combination of key + description (for example, ",").

Key direction
By default defined as LTR (left to right).

Host Keys Manual Definition

Caption
The caption that will appear on the customized host key.

Host key
Determines the host key that will be sent. The host key should be in square brackets and can be selected from the standard host keys list to the right of the Host key field.

Cursor position
Once the host key is sent the cursor is placed in the position defined here. When these fields are left empty, the cursor remains in the current position.

Keyboard Mappings

Use the Keyboard Mappings node to map host keys to keyboard keys.

Keyboard key/s
The combination of keyboard keys such as Shift+W. The values in this field are entered into the field as they are pressed on the keyboard.

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).

Command
Determines the host key that will be sent. The host key should be in square brackets and can be selected from the standard host keys list to the right of the Host key field.

Hexadecimal code (VT hosts only)
A sequence of hexadecimal numbers assigned to each key. Examples for possible values: 0x7F or 0x08 (default in ApplinX). If [backspace] doesn't work as expected in ApplinX, try configuring 0x7F.

Language

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

Language
The application's language.

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 identification properties.

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.

Typing direction
Right-to-left languages may display typed-in text in the client application text fields, aligned to the left of the field. In order to display the text aligned to the right, select the right-to-left option. Numeric type fields typing direction will always be left-to-right.

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.

Navigation

Enable map steps recording while navigating in session
Once selected, the application map will record the navigation steps between the screens.

Exception path
An Exception path is initiated once a Path Procedure which is being executed reaches a screen that is not defined as the To Screen of the step that was executed or as the From Screen of the next step to be executed. When such a screen is reached, ApplinX immediately searches (ignoring the wait condition timeout) for the Exception Path that has been defined. If an Exception path is found, it is executed. ApplinX will then try to continue the original path from the point before the Exception Path was initiated. The attempt to find and execute a suitable Exception path will be repeated as many as 30 times, after which the original path will fail to be executed.

Printer Parameters

ApplinX supports printer sessions only on AS/400 and mainframe hosts. It connects to the host, retrieves the print buffers and analyzes them. The host handles the printer's queue and the connection of printer sessions to display sessions. ApplinX connects to the host as a printer session to receive the print buffers and allows you to work with them.

Note: This is not available when working with a SOA license, and is only available when working with a Web enabled license.

Check the **Enable printer** check box to enable this feature. It is enabled by default when creating a new application.

Device name
The name of the printer device as defined in the host.

Associate
The name of the display device (on the host) that the printer should connect to.

Note: Applicable for mainframes only. The host handles the linkage of printer session to the display device.

Message queue
The message queue where printer messages are sent. The default message queue is contained in library QSYSOPR (enabled for AS/400 hosts only).

Message library
The name of the library that contains the message queue (enabled for AS/400 hosts only).

Delay response to host
Adds a delay before the ApplinX server sends a response to the host, as a number of hosts react in an unpredictable fashion when a very fast response is received from the emulator (e.g. creating problems with the printer session). This parameter should be set only after experiencing such problems.

Create trace file
Indicates whether to log the communication with the host into a file. If you include %u or %t (or both) in the trace file name, you can create files for diverse users with different time information and session ID.

Note: This trace file may only be used as a replay file for a printer session.

File name
The name of the trace file. You can **create a separate file for each** printing session, or connection and the name can include the session ID and/or creation time.

%u will insert the session ID.

%t will insert the time stamp of the connection.

By default, this feature is not enabled.

Open recordings folder
Opens the Windows Explorer and displays the location and list of existing trace files.

Refer to the Printer Session chapter for more information regarding the Printer Session.

Printer>Offline

Work offline
Indicates whether to simulate a communication with the host by using a pre-recorded file. By default, this feature is not enabled.

Files on server
Lists the trace files which are on the server. Click and browse to copy a file from your local machine to the server.

Simulate host delay
Allows 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). Available values: No delay, Simulate host, 500 ms, 1000 ms, 2000 ms, 3000 ms, 4000 ms, 5000 ms, 6000 ms, 7000 ms, 8000 ms, 9000 ms, 10000 ms (by default No delay is selected).

Process Tracing

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 *Software AG\ApplinX\host-applications\<application name>\auditing directory*.

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

Enable process tracing
Select this check 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 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 textual informative format (txt) and/or to display the logged data in 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 connection ID.

%t will insert the time stamp of the connection.

%c will insert the connection ID.

By default, this feature is not enabled.

Open auditing folder
Opens the Windows Explorer and displays the location and list of existing files.

Repository

Since version 9.8, ApplinX uses an internal database based on the H2 database as a repository. External databases are no longer supported as repositories and are migrated to the internal DB configuration.

The Database Manager is a migration tool, allowing you to convert your databases' structure to the current structure needed by the installed version of ApplinX.

Screen Content

Replace padding characters with space
Some hosts fill input fields with a character such as underscores or dots (following the actual content of the field). You may define that ApplinX replaces this character in the input field with spaces when the server sends the field's contents to the client. It is possible to define up to two padding characters.

The radio buttons enable applying this to all fields or only to input (unprotected) fields (input fields also include application fields with "both" protection type - protected and unprotected).

Remove from both sides of text

This field determines whether the padding character will be removed from both sides of the field or just from one side (in LTR applications the character will be removed from the right side, and in RTL applications the character will be removed from the left side). In numeric type input fields, the characters are removed from both sides of the content (as it is clearly not a number).

Trim Fields

Some hosts fill input fields with spaces (following the actual content of the field). You may instruct ApplinX to remove the spaces from the content of the input field.

The radio buttons enable applying this to all fields or only to input (unprotected) fields or input fields and application fields with "both" protection type - protected and unprotected).

Remove from both sides of text

This field determines whether the spaces will be removed from both sides of the field or just from one side (in LTR applications the character will be removed from the right side, and in RTL applications the character will be removed from the left side). In numeric type input fields, the characters are removed from both sides of the content (as it is clearly not a number).

Return Content of Hidden Fields

Some hosts have hidden fields in the screen. Usually these fields' visibility depends on the context of the screen. By default, ApplinX does not return the content of these fields because the user in an emulator cannot see them. However, sometimes their content can be important and one may want to access it (for example, hidden fields can be used as screen identifiers, or as a part of a Path Procedure's logic). Check the **Return content of hidden fields** check box to return the content of all hidden fields in the application.

Split fields when an attribute changes (color, blinking etc.)

This field determines whether when an attribute changes in the middle of a field, this field will be split.

Windows

In the Designer, you can describe the way a window is displayed in the host. The Window definitions are used to correctly identify screens and to open host windows as separate pop-up windows. Click the **Add** or **Delete** button to make any changes.

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.

List of host windows

The list of host windows is displayed here. Add or remove windows by clicking on the relevant button. 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.

Attributes

Frame is intensified
One of the parameters that assist in recognizing windows. This parameter determines whether the frame will be recognized as a window when the frame is intensified.

Frame with title
One of the parameters that assist in recognizing windows. This parameter determines whether the frame will be recognized as a window if the frame has a title.

Identify window even though unprotected fields exist outside window area
This field determines whether or not an area that seemingly could be identified as a window will be recognized as one, even though there are unprotected fields outside the window area. Note: When a window is identified within a screen, and working with modal windows, it is not possible to edit unprotected fields outside the window area.

Content

Remove window frame
Characters appear on the first line of the screen without the window frame. Suitable typically for NATURAL mapping applications.

Display window title
Displays the window title in the window frame to be viewed.

Frame characters

Vertical
The character used for the host's modal window vertical lines. The character can be either typed in, or selected from the list of characters.

Horizontal
The character used for the host's modal window horizontal lines. The character can be either typed in, or selected from the list of characters.

Corners
The character used for the host's modal window corners. The character can be either typed in, or selected from the list of characters.

Synchronize corners
Defines that all four corners of the window frame are the same character (default). When selecting this option, define the character for each of the corners (upper-left, lower-left, upper-right and lower-right.

3 Host Configuration Parameters

Host Configuration: Connectivity

Name/IP address
The host's TCP/IP address (IPv4 and IPv6 address formats are supported).

Port
The TCP/IP port to which the host is listening.

Device type
The device type of terminal that ApplinX emulates.

Protocol
The protocol used to access the host.

Device Type	Protocol	Comment
Mainframe: IBM-3278, IBM-3279 (only supported via SNA servers that convert SNA to TN3270)	TN3270 TN3270E	
Natural-UNIX (Software AG)	Natural-APX, Natural-NSW	
BS2000	9750	
UNISYS-TD830 (UNISYS T27 EBCDIC), UNISYS-TD830-ASCII (UNISYS T27 ASCII)	TELNET	
TANDEM: TN6530-8	TN6530	Only block mode applications are supported in Web enablement.
FUJITSU-6680-00, FUJITSU-6680-02	TN6680	
AS/400: IBM-3179, IBM-3477-FC, IBM-5555	TN5250 TN5250E	
HITACHI	TN560	Supported only by ApplinX Servers installed on Win 32 platforms.
VT	VT-100, VT 220-7, VT 220-8	VT is supported in a SOA enablement only.

Model
The number of characters per column and per row, in the host's window.

Application
The name of the application on the BS2000, to which you want to connect (relevant only to BS2000). The name can be up to 8 characters long. $DIALOG is provided as the default name. The connection to the host will be established using an Open command, without any parameters.

Application script
The name of the shell script file required to start the Natural application (relevant Natural-UNIX).

Parameter file
This file contains the configuration parameters relevant for Natural-UNIX. Enter the parameter file name as it appears in Natural, without the file extension. For example: SYSTRANS.SAG should be written as SYSTRANS.

Connection timeout
The number of milliseconds the application will try to connect to the host, before it announces failure.

Automatically attempt to reconnect
Determines that ApplinX should automatically attempt to reconnect after connection failure. This option is deprecated and will be removed in the next version. We suggest you use ApplinX Connection Pool functionality instead. A connection pool allows you to reconnect a session based on the pool's policy. See *Connection Pools*.

Code page
The code page number of the required language for this application.

Convert input to uppercase
Sends data to the host as uppercase (overrides default host configuration-backwards compatible).

Use 8-bit data
Determines whether to use an 8-bit ASCII table. When not selected, the 7-bit ASCII table is used by default.

Auto wrap
This option is available for VT hosts only. Selecting this option starts a new line when the current line of characters reaches the margin.

Options

This is not available for AS/400 hosts.

Enable Natural-Data-Transfer support
Enables uploading and downloading Natural data transfer files.

Ignore leading form feed in downloaded report
When you are downloading a report, an empty page precedes the report. However, if this check box is selected, this empty page is not generated. This appliea to reports generated using the Natural WRITE command (available for Mainframe hosts only).

Keep trailing blanks at the end of downloaded records
If this check box is selected, the trailing blanks are also written to disk. This option applies to reports generated using the Natural WRITE WORK command (available for Mainframe hosts only).

RPC

Available in SOA applications only.

Note: This tab is not displayed in Mainframe host applications.

Username
Insert a User ID to log into an AS/400 host.

Password
A password to log into an AS/400 host.

Library list
A list of libraries separated by spaces, in the AS/400 host called by programs that may reference them. If a program calls a library that is not recorded in the Library list, the program may not function properly.

Create debug log
Check this check box to save a debug log when running ApplinX programs.

File name
Specify the log file.

Security

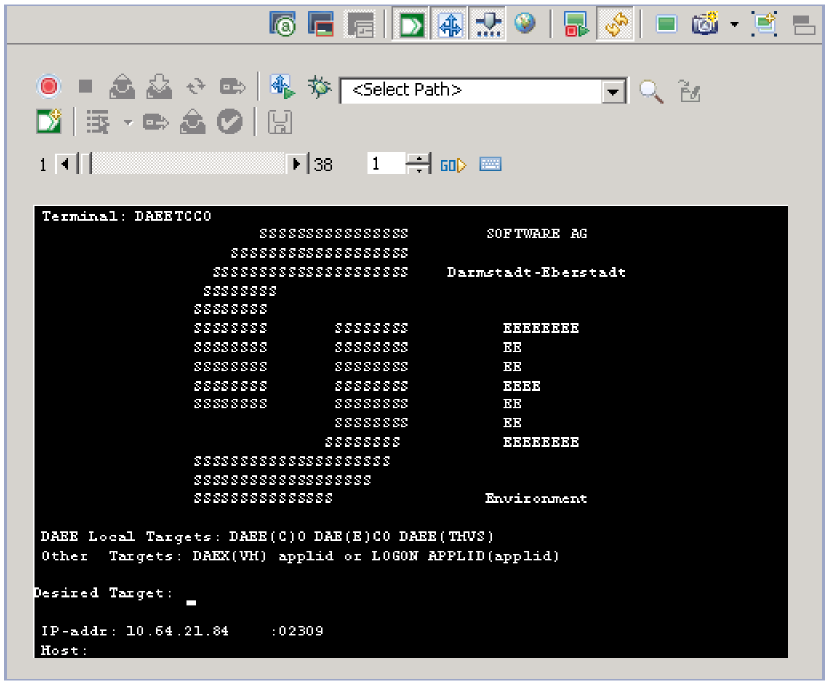
Connect using SSH
Enables connecting using the SSH connection between the ApplinX server and the host.

Use SSL connection to host
Enables using the SSL connection between the ApplinX server and the host.

Add and Remove icons
The Add icon enables adding a valid X509 certificate. Use the remove button to remove certificates not used.

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

4 Display Session Properties and View



```
 Terminal: DAEETCCO
                        SSSSSSSSSSSSSSSS         SOFTWARE AG
                     SSSSSSSSSSSSSSSSSSS
                   SSSSSSSSSSSSSSSSSSSSS     Darmstadt-Eberstadt
                  SSSSSSSS
                 SSSSSSSS
                 SSSSSSSS       SSSSSSSS           EEEEEEEE
                 SSSSSSSS       SSSSSSSS           EE
                 SSSSSSSS       SSSSSSSS           EE
                 SSSSSSSS       SSSSSSSS           EEEE
                 SSSSSSSS       SSSSSSSS           EE
                                SSSSSSSS           EE
                               SSSSSSSS            EEEEEEEE
                 SSSSSSSSSSSSSSSSSSSSS
                 SSSSSSSSSSSSSSSSSSS
                 SSSSSSSSSSSSSSS                 Environment

 DAEE Local Targets: DAEE(C)0 DAE(E)C0 DAEE(THVS)
 Other  Targets: DAEX(VH) applid or LOGON APPLID(applid)

Desired Target: _

 IP-addr: 10.64.21.84    :02309
 Host:
```

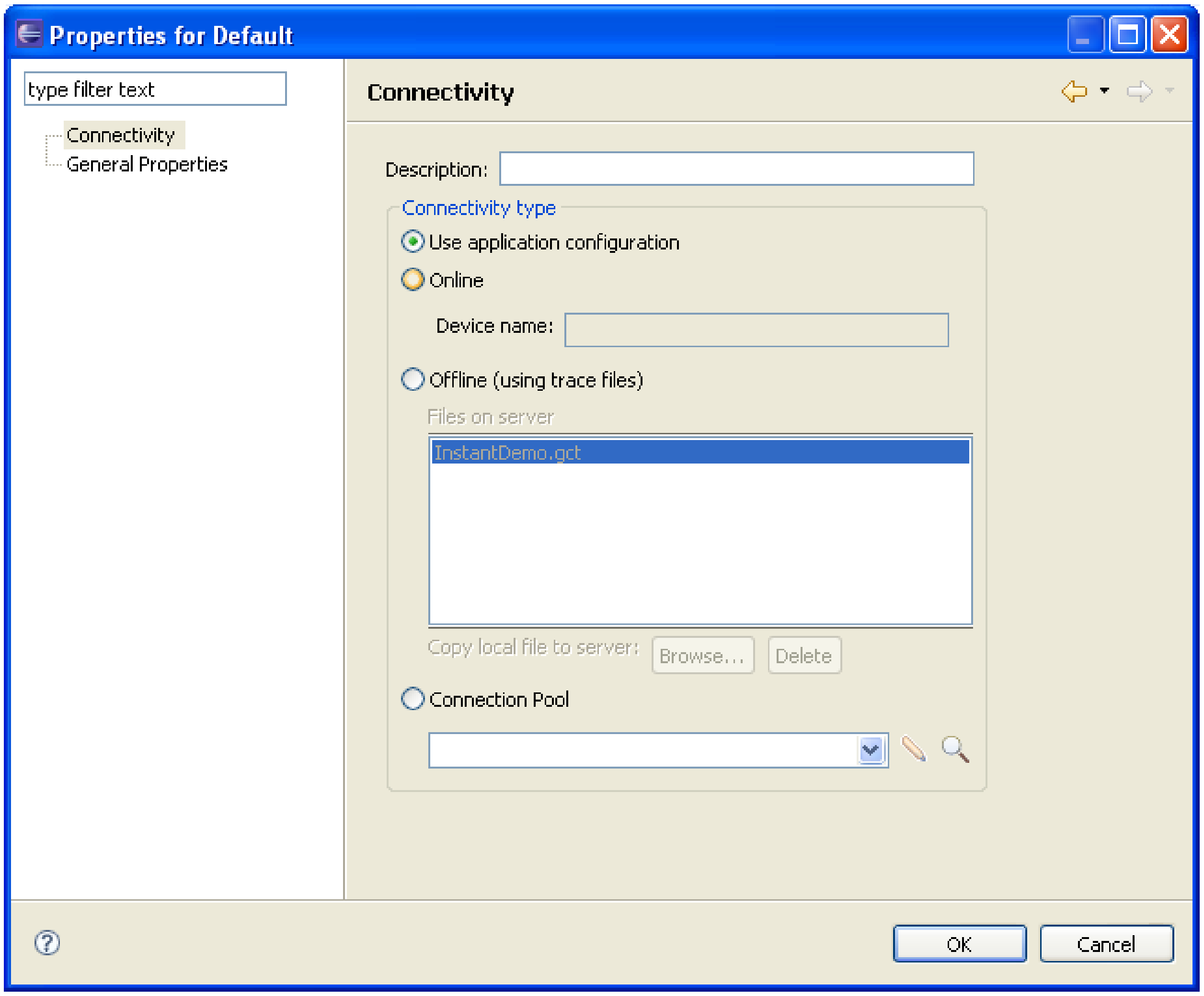
Display Session Properties

Refer to Creating a Display Session. These definitions override the default connection configuration as defined in the application properties.

- Connectivity Properties

- General Properties

Connectivity Properties



Description:

Connectivity type

(●) Use application configuration

( ) Online

Device name:

( ) Offline (using trace files)

Files on server

InstantDemo.gct

Copy local file to server: Browse... Delete

( ) Connection Pool

Description
A suitable description for the session.

Use application configuration
Select Use application configuration to implement the configuration as set in the application properties.

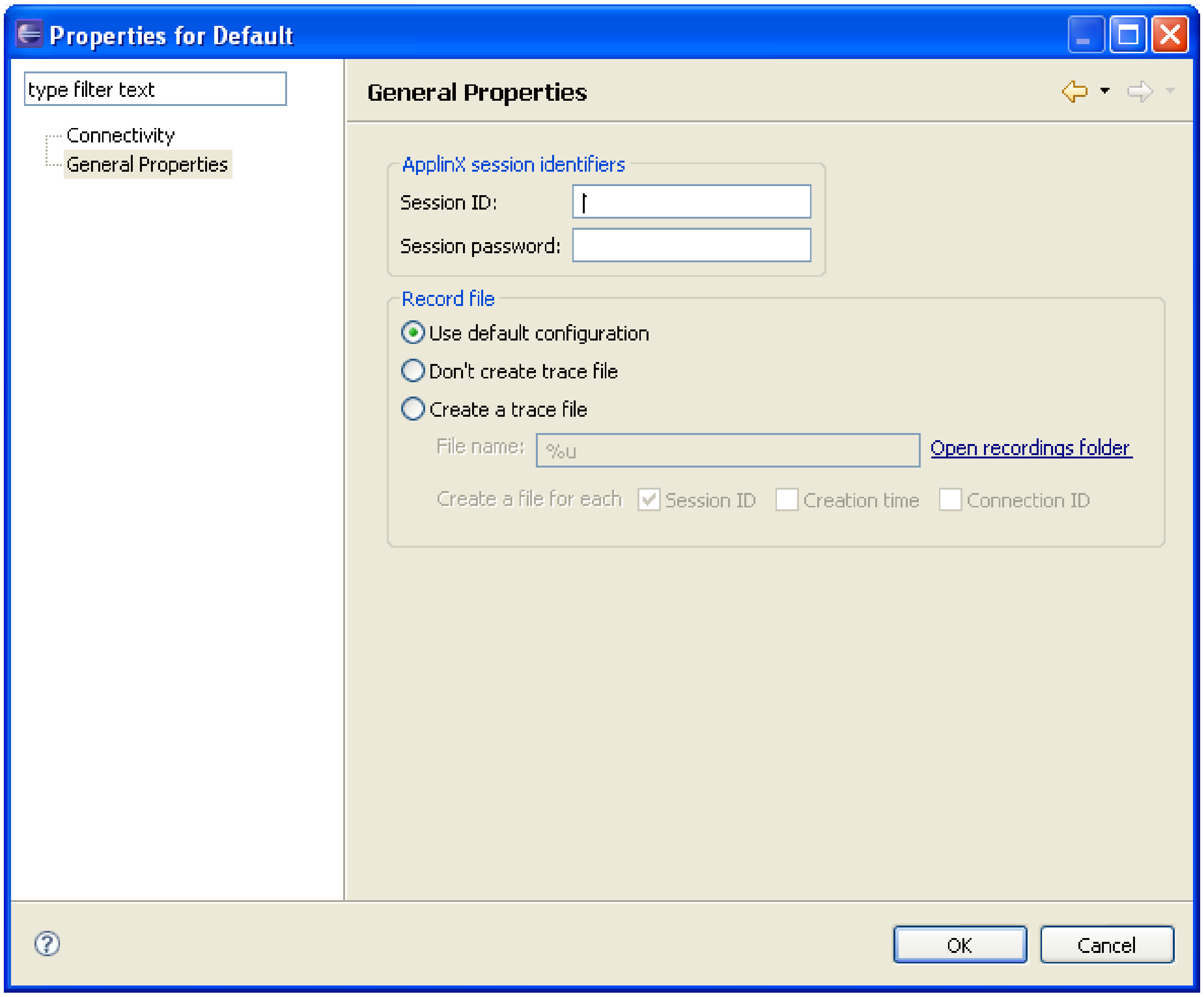
Online
Select Online when you want to connect online to the application's configured host.

Offline (using trace files)
Indicate whether to simulate a communication with the host by using a pre-recorded file. Select Offline (using trace files) and then select the replay file from the list displayed in the **Files on server** list or browse and select a file from a folder.

Connection Pool

Select Connection Pool and choose one of the application's Connection Pools.

General Properties



Session ID

Custom session ID

Session password

Custom session password.

Use default configuration

Select Use default configuration to implement the Application Trace File configuration as set in the application properties.

Don't create trace file

Select Don't create trace file if you do not want to create a trace file.

Create a trace file

Indicates whether to log the communication with the host into a file. Select Create a trace file and insert the trace file name. If you include %u or %t (or both) in the File name, you can create files for diverse users with different session ID and time information. Check Session ID if you want to override the files of the same session ID. Check time to add a time stamp to the file name. This does not override previous files. Check Connection ID to add full connection ID (application, connection pool, connection) to the file name.

Session View Main Toolbar

	Show/hide attributes
	Show/hide input fields
	Window: will gray out or display in regular colors, the screen area outside the window. It is not possible to navigate outside the window.
	Business Activity
	Path Toolbar
	Navigate to Screen Toolbar
	Show HTML Preview
	Restart Session
	Synch with host
	Character mode/Block mode
	Change screen direction
	Update Screen Image
	Edit the current screen
	Identify new screen/screen group
	Automatically identify unprotected screens
	Screen creation mode: automatic, semi-automatic or manual.

Business Activity Toolbar

	Capture business activity
	Select rule
▶	Start rule
	Participating Screens Rule
✗	Abort Rule
	Complete Rule
	Mark Fields to Check
	Mark Fields to Map for Output
✓	Add Rule
	Save a Business Entity

Path Procedure Toolbar

The Record Procedure Toolbar is used to record, run and debug a Path Procedure.

●	Record
■	Stop
	Marks the input (unprotected) fields of the current screen
	Marks the procedure outputs of the current screen
	Loop
	Get loop condition text
▶	Run
	Debug

Application Map Toolbar

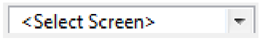
The Application Map toolbar is available to be used when developing an application. Using the toolbar you can:

- Check that the Application Map definitions work as expected (by selecting a screen, and checking that ApplinX is able to navigate to this screen).
- Navigate to a specific screen to edit and make changes in the screen.

The Application Map toolbar is not displayed when working offline.

	Navigate to screen
	Open the entity's editor
	Find entity

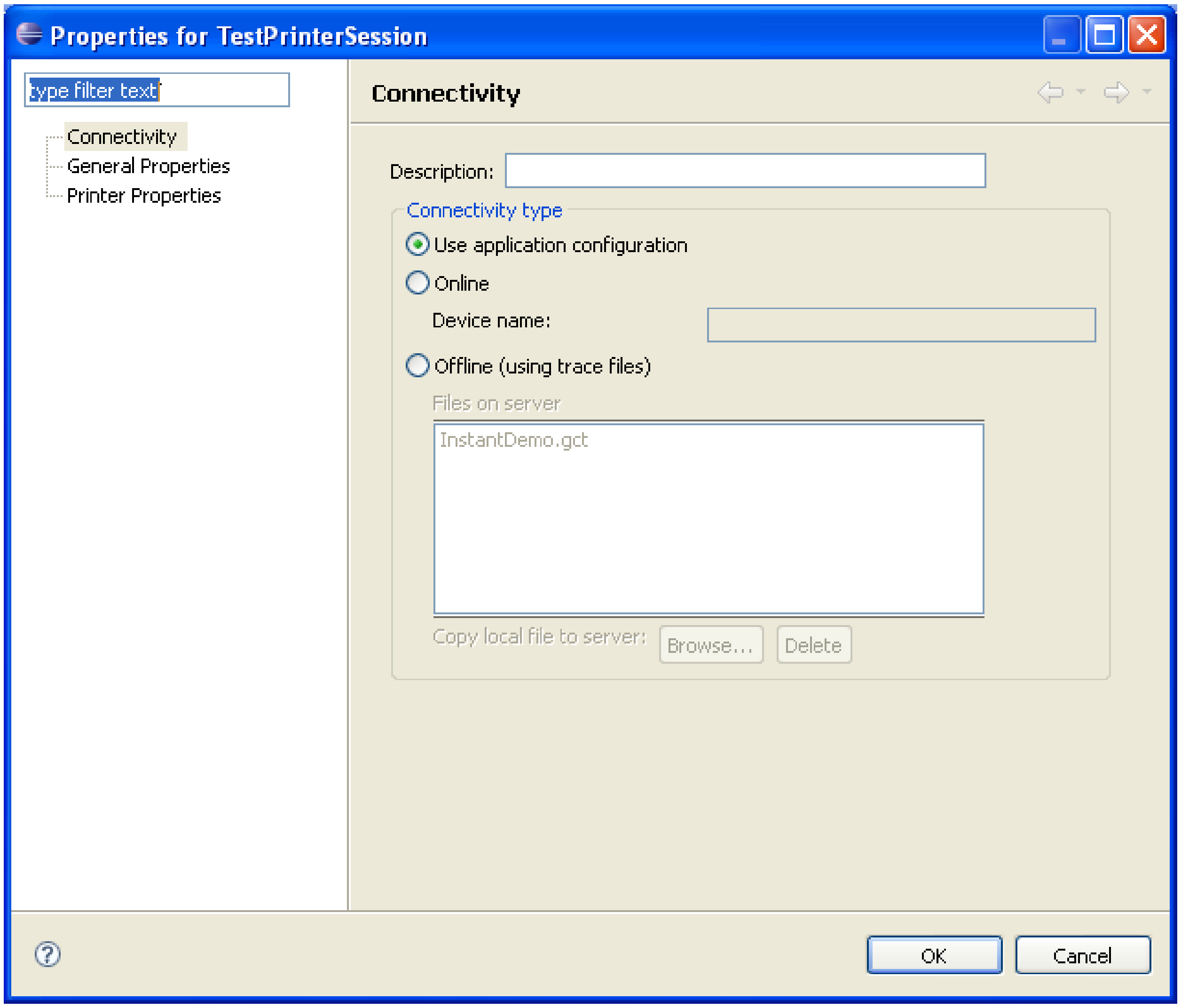
Navigation Toolbar

	Slider, indicates current location in the replay file.
	Screen number box, displays the current screen number.
	Displays a list of the available screens and screen groups.
	Open the entity's editor
	Find entity
	When clicked, displays the screen number entered in the screen number box.
	Show User Input button displays the user input entered while the replay file was recorded. 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.

5 Printer Session Properties

Connectivity



Description:

Connectivity type

(● Use application configuration) ○ Online ○ Offline (using trace files)

Device name:

Files on server

InstantDemo.gct

Copy local file to server:

Description

A suitable description for the session.

Use application configuration

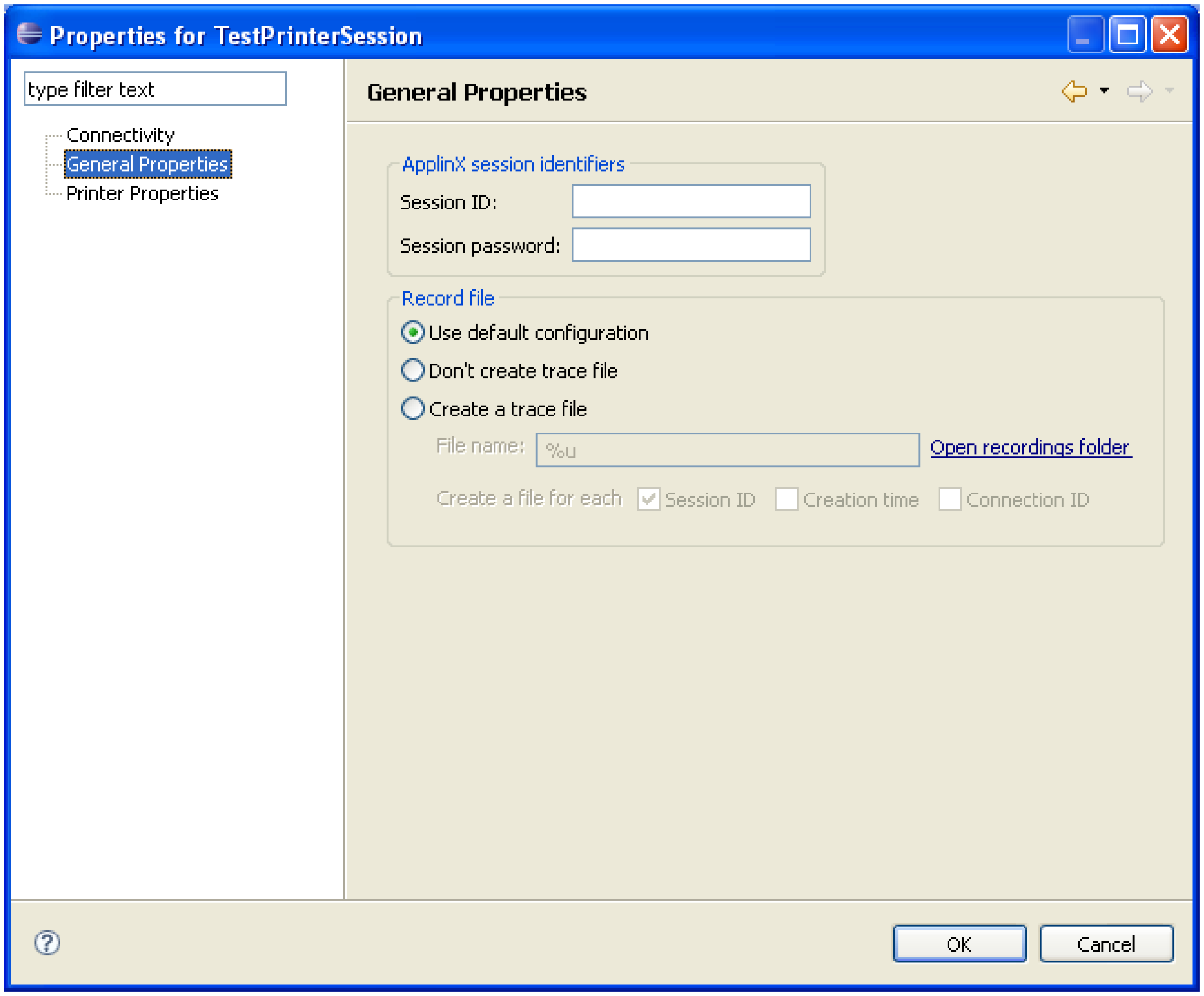
Select Use application configuration to implement the offline configuration as set in the application printer properties.

Online

Select Online when you want to connect online and do not want to use a trace file. Enter a device name.

Offline (using trace files)
Indicate whether to simulate a printer session by using a pre-recorded file. Select Offline (using trace files) and browse and select a replay file from a folder.

General Properties



General Properties

ApplinX session identifiers
Session ID:
Session password:

Record file
(●) Use default configuration
( ) Don't create trace file
( ) Create a trace file
File name: %u
Create a file for each [✓] Session ID [ ] Creation time [ ] Connection ID

Session ID
Custom session ID

Session password
Custom session password.

Use default configuration
Select Use default configuration to implement the Application Trace File configuration as set in the application properties.

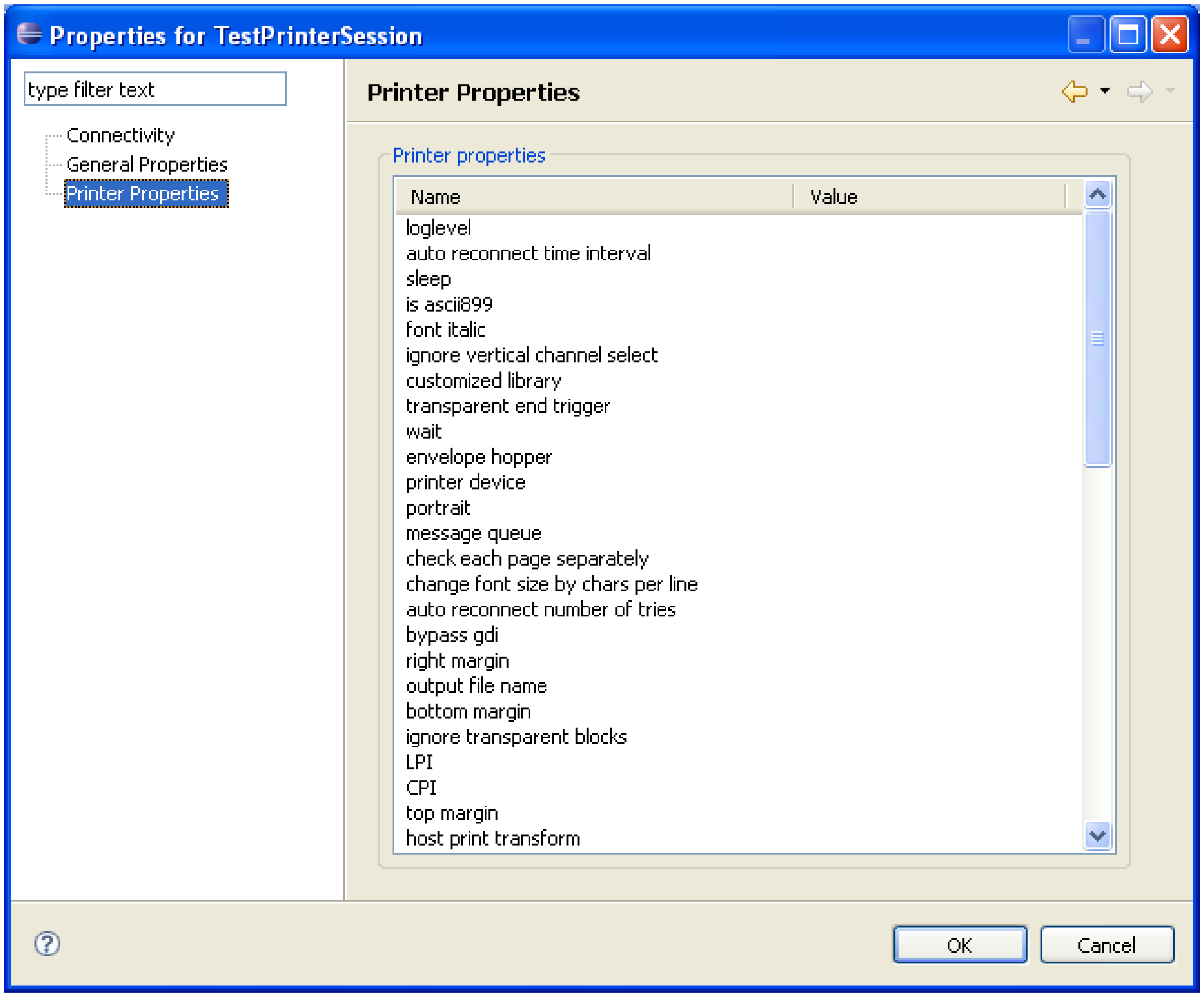
Don't create trace file

Select Don't create trace file if you do not want to create a trace file.

Create a trace file

Indicates whether to log the communication with the host into a file. Select Create a trace file and insert the trace file name. If you include %u or %t (or both) in the File name, you can create files for diverse users with different session ID and time information. Check Session ID if you want to override the files of the same session ID. Check time to add a time stamp to the file name. This does not override previous files. Check Connection ID to add full connection ID (application, connection pool, connection) to the file name.

Printer Properties



Name	Value
loglevel	
auto reconnect time interval	
sleep	
is ascii899	
font italic	
ignore vertical channel select	
customized library	
transparent end trigger	
wait	
envelope hopper	
printer device	
portrait	
message queue	
check each page separately	
change font size by chars per line	
auto reconnect number of tries	
bypass gdi	
right margin	
output file name	
bottom margin	
ignore transparent blocks	
LPI	
CPI	
top margin	
host print transform	

Enter values in the value column. Using the Printer Properties it is possible to set and test various printing jobs

II ApplinX Entities

6 Screens, Fields and Mappings

In the Screen Editor edit the screen definitions: the identifiers, fields, transformations, screen groups, tables and map steps. The Screen Editor can be accessed after creating a new screen or by editing an existing one. Refer to Screens.

Screens/Screen Groups

Identifiers Tab

Lists the identifying elements that identify the screen. Refer to Identifiers, Adding or Deleting an Identifier and Editing a Screen Identifier.

Identify entire screen (ignore window definition)

Selecting this check box indicates that the application's window definitions are ignored during the identification of this screen. Once the screen is identified, the full screen is used. This check box is only enabled when the screen has a window.

Select this check box:

- When the screen includes multiple windows and the window definition in this screen is ambiguous.
- To override general application window identification definitions in a specific screen.

Note: Changing the selection of this field will cause existing position dependant definitions in the screen (such as identifiers, application fields and transformation regions) to be incorrect. It is therefore necessary to redefine identifiers, application fields and transformation regions when selecting/clearing the check box. By default this field is not selected.

List of Identifiers

Lists the existing identifiers. There are several columns in this table:

Icon	Graphic indication of the type of identifier.
Type	Textual indication of the type of identifier
Content	Description of the identifier contents
Row	The position within the screen
Column	The position within the screen

Application Fields Tab

Lists the screen's mappings. Enables adding new mappings, such as single, multiple or dynamic mappings. Next to each mapping, an icon indicates whether the mapping is inherited from a screen group, whether the mapping has been defined locally for the current screen, or whether the mapping was originally inherited from a screen group but has been overridden locally to suit the current screen. 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.

Icon	Description
◻	The mapping has been defined locally for the current screen.
↳	The mapping is inherited from a screen group.
◳	The mapping was originally inherited from a screen group but has been overridden locally to suit the current screen .
✖	Cancels overriding an inherited mapping.

Icon Legend

The Override/Cancel Override hyperlink (enabled when not selecting a local mapping) overrides an inherited mapping, or cancels overriding an inherited mapping (using the inherited values).

The Add Field Mapping hyperlink enables adding a new field mapping.

The Delete Field Mapping hyperlink (enabled on when selecting a local mapping) removes the mapping.

VT Parameters within Field Mappings Tab

Override default parameters
Overrides VT screen parameters for specific mappings in the current screen. 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.

Automatic skip
When checked, indicates that the cursor skips automatically to the next input field upon reaching the end of the current field (happens when the field is completely filled with text).

Skip one position
When checked, indicates that the cursor skips automatically when the field is completely filled, but it skips only one position and not to the next field.

Always send this field
As ApplinX operations may not work in an efficient manner when working with these hosts (require cursor movements), ApplinX attempts to perform a minimal set of operations, thus preferring to send only fields that have different content than what was returned from the host. In very rare cases, the user keystrokes need to be sent even when they do not change the original content. In these instances **Send all fields** should be selected.

Override default unprotected field navigation
Indicates that the tab movement will be different from the default tab movement.

Next mapping
Defines the next mapping the tab will move to.

Index
For multiple mappings, moves to the Next mapping according to the index.

Host field behavior
This property indicates the behavior of a certain field (alignment of the text and cursor for a certain field, hidden for password field etc.). The ApplinX Server uses it in order to place sent text in the correct position in the field, and waits for the correct text in the correct position after sending it. The default value is blank (the text is sent in the place where the cursor is positioned, and the contents received from the host are not validated).

- Left: Cursor and text are aligned to the left-most position of the field (default).
- Right, zero fill: The cursor and text are aligned to the right. Leading zeros are prepended to the text.
- Right, blank fill: The cursor and text are aligned to the right. Blanks are prepended to the text.
- Right, cursor fixed to the right: The cursor stays fixed in the right-most position of the field as the user types in the field.
- Right, cursor fixed to the right with offset: The cursor stays fixed to the right of the field, but with a certain offset from the rightmost position (usually the offset is 1). Offset text field: the number of host columns offset of the cursor position from the rightmost position of the field.
- Left, cursor fixed to the left with offset: Similar to the above, but the cursor is fixed to the left of the field, with a certain offset to the left.
- Right, cursor left: The cursor is aligned to the left while typing text in the field, but after leaving the field, the text is aligned to the right.
- Hidden: Do not check - the text typed in the field does not appear exactly as typed after leaving the field (applies to password fields and to special format fields like currency fields).
- Hidden, cursor moves: Text does not appear in the field after typed, but the cursor automatically skips to the next field if the field was completely filled.

Uppercase
Determines whether to override the host configuration (defined to send the data converted to uppercase).

Transformations Tab

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

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. Refer to the relevant Transformation topics for further details.

Icon	Description
▣	The transformation has been defined locally for the current screen.
▣↘	The transformation is inherited from a screen group.
☒↘	Disables the inherited transformation.
☒	Disables the local transformation.

Icon Legend

The Enable/Disable Transformation hyperlink allows you to determine whether to use the selected inherited transformation or not.

The Add Transformation Mapping icon enables associating an existing transformation with this screen.

The Delete Transformation Mapping hyperlink (enabled on when selecting a local transformation) removes the transformation from the list.

Screen Groups Tab

Enables associating screen groups from the list of available screen groups to the current screen (relevant for Screens only)

For more information, refer to Screen Group Tasks.

VT Parameters

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. For details regarding each of these parameters refer to the VT Application parameters.

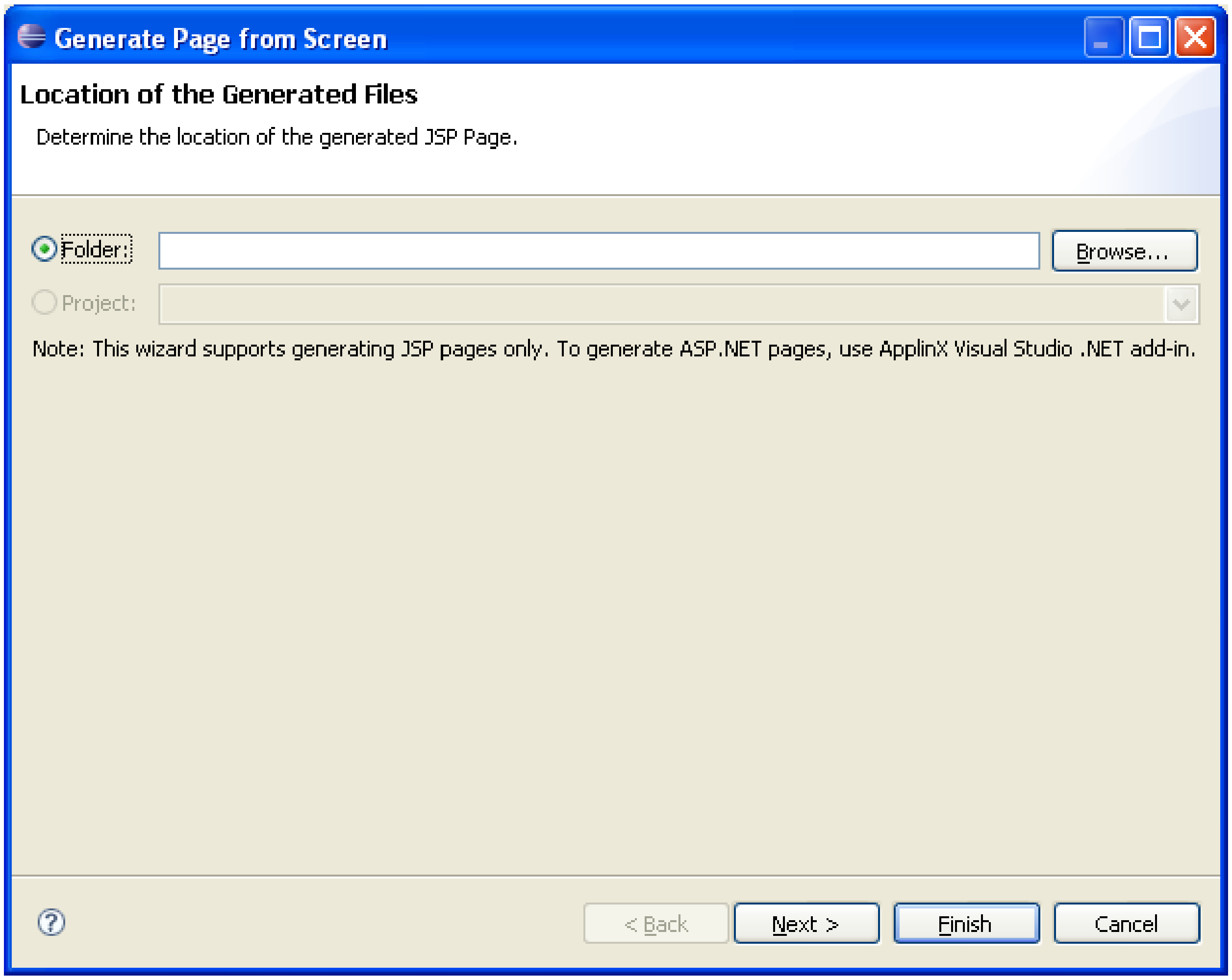
Parameters for Generating JSP Page from Screen

It is possible to generate a JSP page from a screen (refer also to Generating a JSP/ASPX Page from a Screen).

- Location of Generated Files
- Screen Contents

- Code Snippets

Location of Generated Files



Generate Page from Screen

Location of the Generated Files

Determine the location of the generated JSP Page.

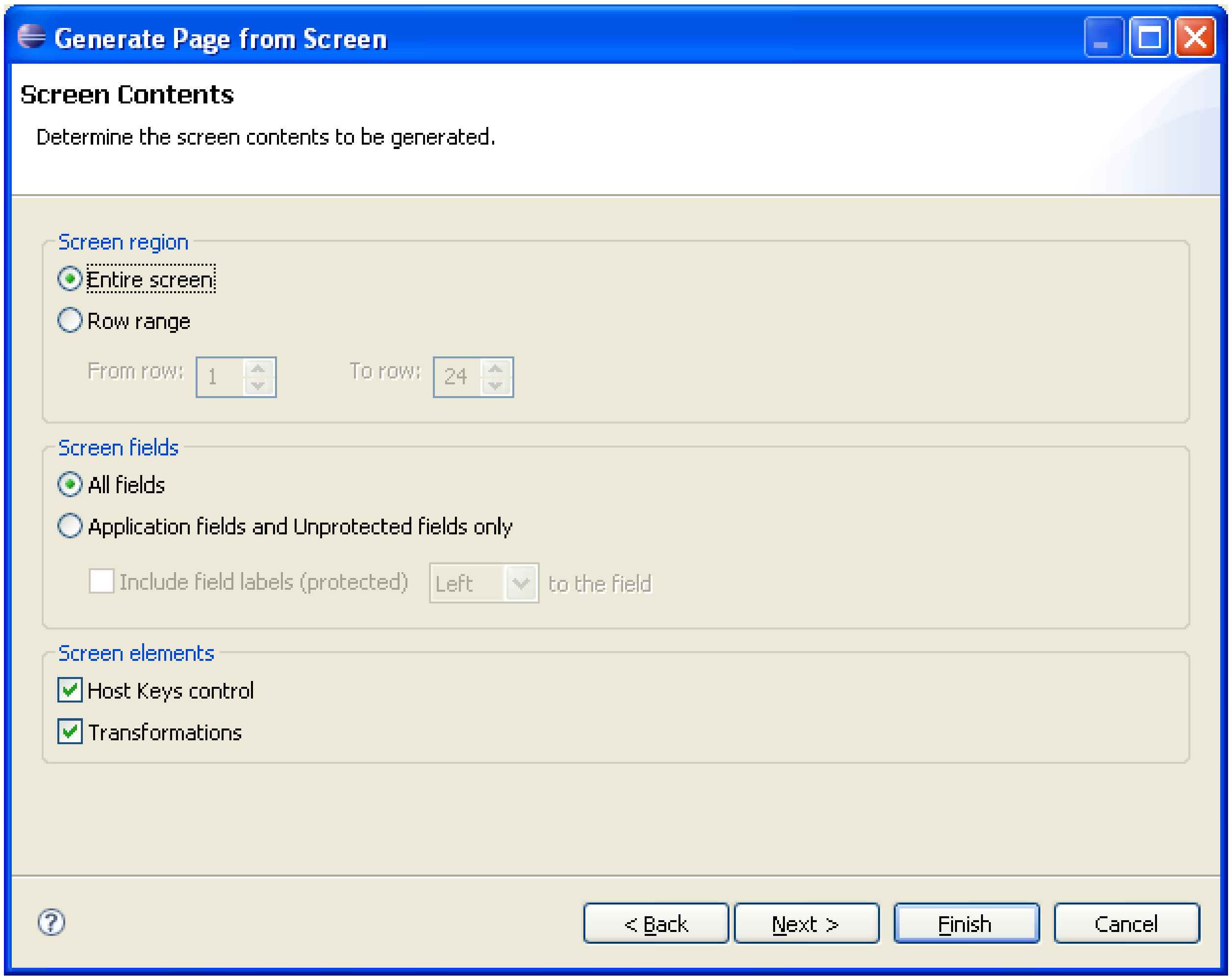
[✓ Folder:] ____ Browse...

Project:

Note: This wizard supports generating JSP pages only. To generate ASP.NET pages, use ApplinX Visual Studio .NET add-in.

Indicates the folder where the generated files will be placed.

Screen Contents



Generate Page from Screen

Screen Contents

Determine the screen contents to be generated.

Screen region

[✓ Entire screen]

Row range

From row: 1 To row: 24

Screen fields

[✓ All fields]

Application fields and Unprotected fields only

Include field labels (protected) Left to the field

Screen elements

[✓ Host Keys control]

[✓ Transformations]

Screen region

Determines the generated area: either the entire screen or a rectangular area.

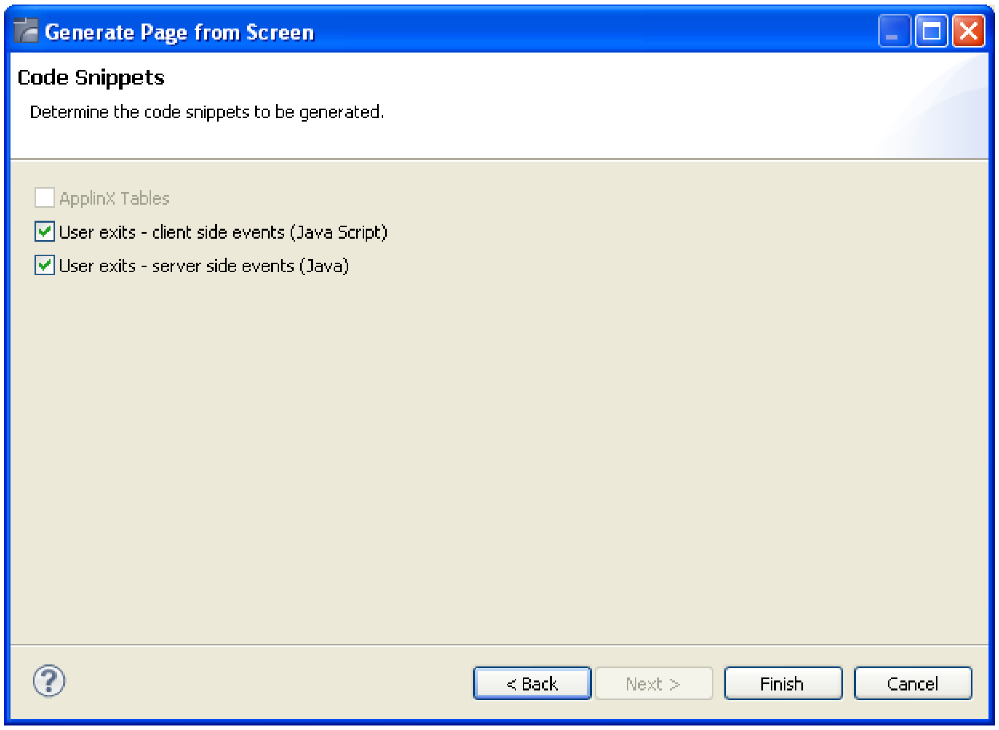
Screen fields

Determines whether to generate all the types of fields or only application fields and unprotected fields (and not generate protected and output fields). If you select to generate application fields and unprotected fields only, then determine whether to generate the input fields without labels or with the label that appears to the left of the input field or with the label that appears to the right of the input field (Include field labels left/right to the field).

Screen elements

Determine whether to generate the Host keys control and/or transformations.

Code Snippets



Code Snippets

Determine the code snippets to be generated.

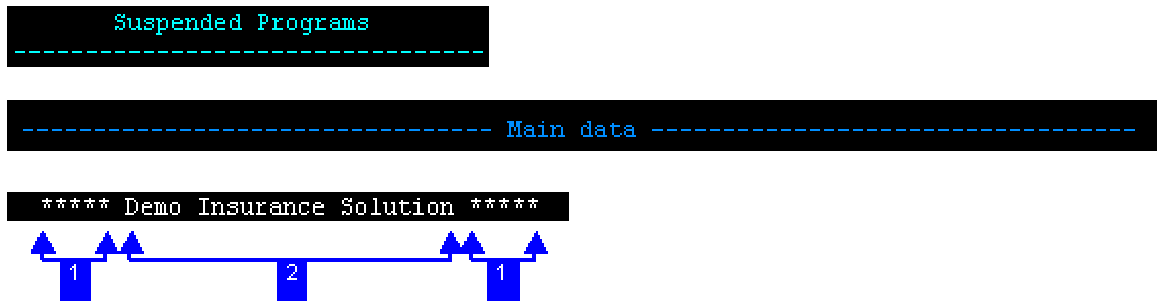
- [ ] ApplinX Tables
- [x] User exits - client side events (Java Script)
- [x] User exits - server side events (Java)

In the Code Snippets tab define the elements you would like to generate code behind for: code transformations, ApplinX host windows, user exits - client side events (JavaScript) or user exits - server side events (Java). The code class contains methods which can be used as a basis for further development.

7 Transformations

Host Pattern Elements

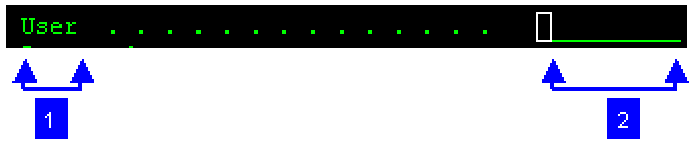
Line Pattern



```
        Suspended Programs
 ---------------------------------

 -------------------------------- Main data ----------------------------------

 ***** Demo Insurance Solution *****
```

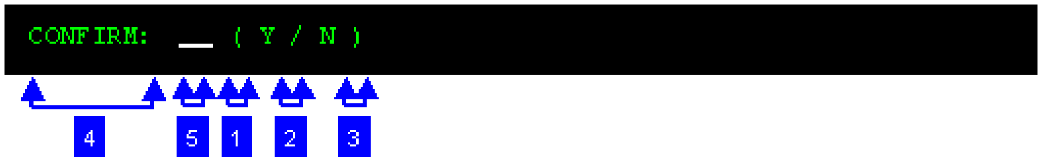
1. Repeated string (*) + minimum subsequent occurrences (5).
2. Title (checked).

Input Field Pattern



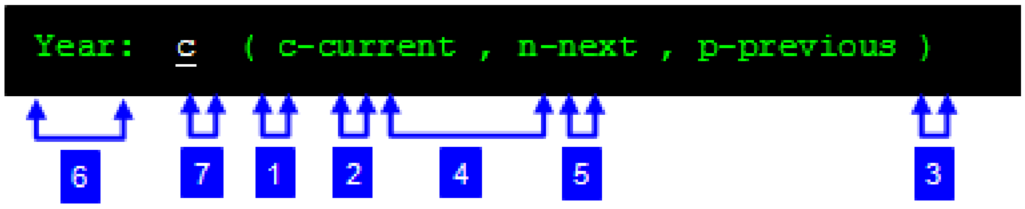
1. Accompanying text (User).
2. Input field's length (8).

Input Field with Values Pattern



```
CONFIRM:  __  ( Y / N )
```

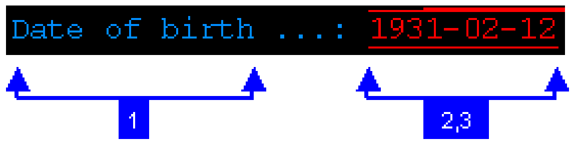
1. Prefix (a left parenthesis '(')
2. Separator (a forward slash '/')
3. Suffix (a right parenthesis ')')
4. Accompanying text (CONFIRM)
5. Input field's length (1)



```
Year:  c  ( c-current , n-next , p-previous )
```

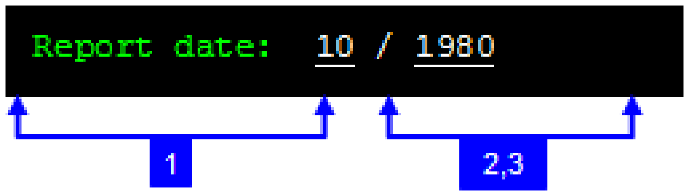
1. Prefix (a left parenthesis '(')
2. Separator (a comma ',')
3. Suffix (a right parenthesis ')')
4. Includes description (checked)
5. Delimiter (a hyphen '-')
6. Accompanying text (Year)
7. Input field's length (1)

Date Input Field Pattern



```
Date of birth ...: 1931-02-12
```

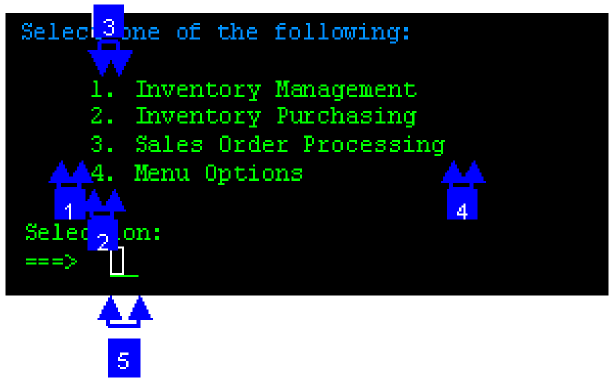
1. Accompanying text (Date of birth)
2. Date format (single input field)
3. Date pattern (yyyy-MM-dd)



```
Report date:  10 / 1980
```

1. Indicating text (Report date)
2. Date format (two input fields)
3. Date pattern (MM/yyyy)

Menu Pattern


```
Select one of the following:

     1. Inventory Management
     2. Inventory Purchasing
     3. Sales Order Processing
     4. Menu Options

Selection:
===>
```

1. String before leading token (spaces). 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.
2. Maximum length leading token (1).
3. Delimiter (a period '.'). 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. String after list item (spaces). Type in the string that appears after the menu item. By default the value of this field is two spaces.
5. Selection field (current cursor position).

Transformation Variables and their Attributes

Throughout the Transformation wizard screens, there are a number of fields whose values you can set using variables. The possible available variables that can be used are determined according to the original pattern. There are two types of variables: text variables such as `$(text)`, `$(line)`, `$(title)` and numeric variables such as `$(text.length)`, `$(line.column)`. You can automatically complete the field to the first variable that matches the text by pressing CTRL+SPACE after ""$(..."". The following table describes the different variables and attributes available for each type of transformation.

Variable	Description	Available Attributes
$(screen.width) $(screen.height)	Used with any of the transformations. Can be useful when positioning elements.	NA

General

Variable	Description	Available Attributes
$(line)	The original repeated pattern.	length, row, column
$(title)	The original pattern's title, when a title was part of the pattern.	length, row, column

Repeating character transformations:

Variable	Description	Available Attributes
$(text)	The original text contents of the text pattern.	length, row, column

Text transformations:

Variable	Description	Available Attributes
$(text)	The original accompanying text of the pattern.	length, row, column
$(input)	The original field input - when relevant, such as a default value of a field.	length, row, column

Input field transformations:

Variable	Description	Available Attributes
$(action)	The value that is sent as the input. This value indicates the action.	NA
$(display)	The value that describes the action, this is normally the text that is displayed.	NA
$(text)	The original accompanying text of the pattern.	length, row, column
$(input)	The original field input when such an input exists (such as a default value of a field).	length, row, column

Input field with values transformations:

Variable	Description	Available Attributes
$(action)	The token value that is sent as the input. This value indicates the action.	NA
$(display)	The value that describes the action, this is normally the text that is displayed.	NA

Menu transformations:

8 Procedures

What are Procedures and Procedure Groups?

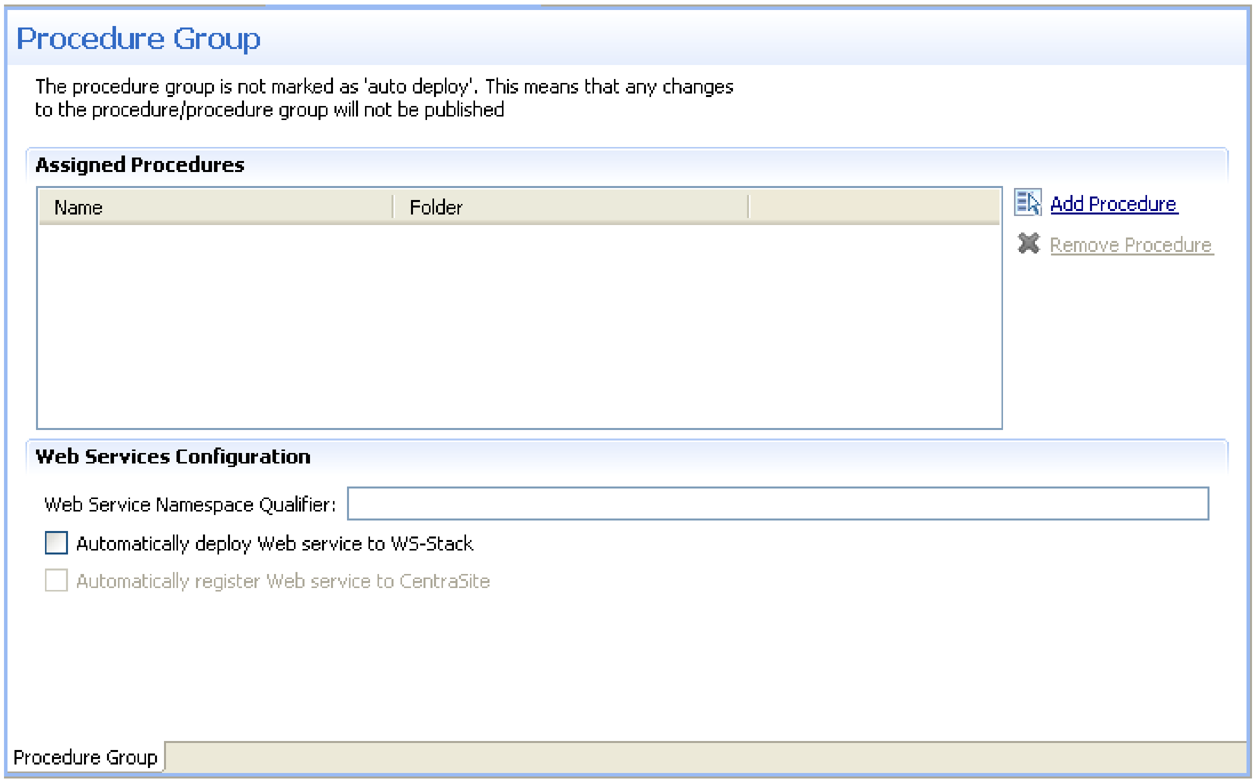
An ApplinX Procedure is a well-defined encapsulation of a complete process, and contains process input arguments, process output arguments and the process definition itself. A procedure group is a container of several procedures. In ApplinX, Procedure Groups are equivalent to Web services, and Web methods 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.

Reference Topics	Related Links
■ The Procedure Group Editor	■ Creating a New Procedure Group
■ The Flow Procedure Editor	■ Registering a Web Service to CentraSite
■ The Path Procedure Editor	■ Deploying a Procedure Group to WS-Stack
■ Procedure Input and Output Attribute Types	■ Assigning a Procedure to a Procedure Group
■ General Nodes (Relevant for both Flow and Path Procedures)	■ How to know which Type of Procedure to use?
■ Flow Procedure Nodes	■ Creating a Flow Procedure
■ Path Procedure Nodes	■ Creating a Path Procedure
■ Web Procedure Nodes	■ Creating a new Web Procedure
■ General Expressions (relevant for Flow, Path and Web Procedures)	■ Defining Procedure Inputs and Outputs
■ Path Procedure Expressions	■ Working with Procedure Nodes
■ Web Procedure Expressions	■ Using the Mapper to Map Source Elements to Target Elements
■ What is an XPath?	■ Debugging Procedures
■ Troubleshooting Web Integration	■ Path Procedure Failure Log
■ Program Procedures	■ Program Procedures

Procedure Groups


Procedure Group

The procedure group is not marked as 'auto deploy'. This means that any changes to the procedure/procedure group will not be published

Assigned Procedures

Name	Folder

Web Services Configuration

Web Service Namespace Qualifier:

☐ Automatically deploy Web service to WS-Stack

☐ Automatically register Web service to CentraSite

In the Procedure Group Editor add and remove procedures relevant to this Procedure Group and define the Web Services Configuration.

Refer to Registering a Web Service to CentraSiteand Deploying a Procedure Group to WS-Stack.

Procedure Input and Output Attribute Types

Attribute Types

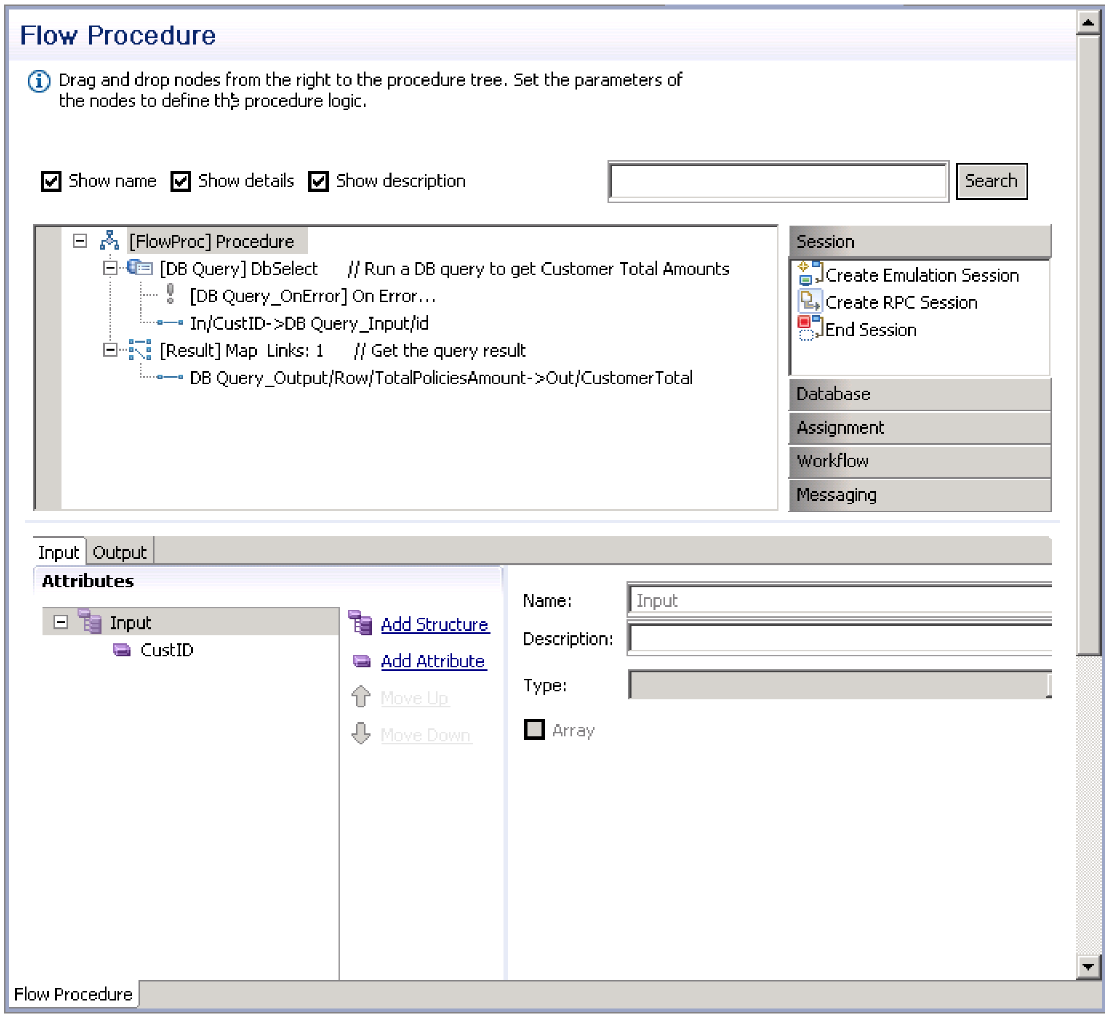
- Text
- Long
- Boolean (True, False)
- Double
- Integer
- Float

- Byte
- Date: The format can be either 2001-07-04 12:08:56.235 or 2001-07-04T12:08:56.235 (this format should be used for Web Services only), or dd/MM/yyyy

Refer to Defining Procedure Inputs and Outputs.

Flow Procedures

Available in SOA applications only.


Flow Procedure

Drag and drop nodes from the right to the procedure tree. Set the parameters of the nodes to define the procedure logic.

☑ Show name ☑ Show details ☑ Show description

```
[FlowProc] Procedure
  [DB Query] DbSelect     // Run a DB query to get Customer Total Amounts
    [DB Query_OnError] On Error...
    In/CustID->DB Query_Input/id
  [Result] Map  Links: 1     // Get the query result
    DB Query_Output/Row/TotalPoliciesAmount->Out/CustomerTotal
```

Session: Create Emulation Session, Create RPC Session, End Session

Name: Input

Description:

Type:

☐ Array

Show name
Includes the name of the node in the procedure details.

Show details
Includes details of the node in the procedure details.

Show description

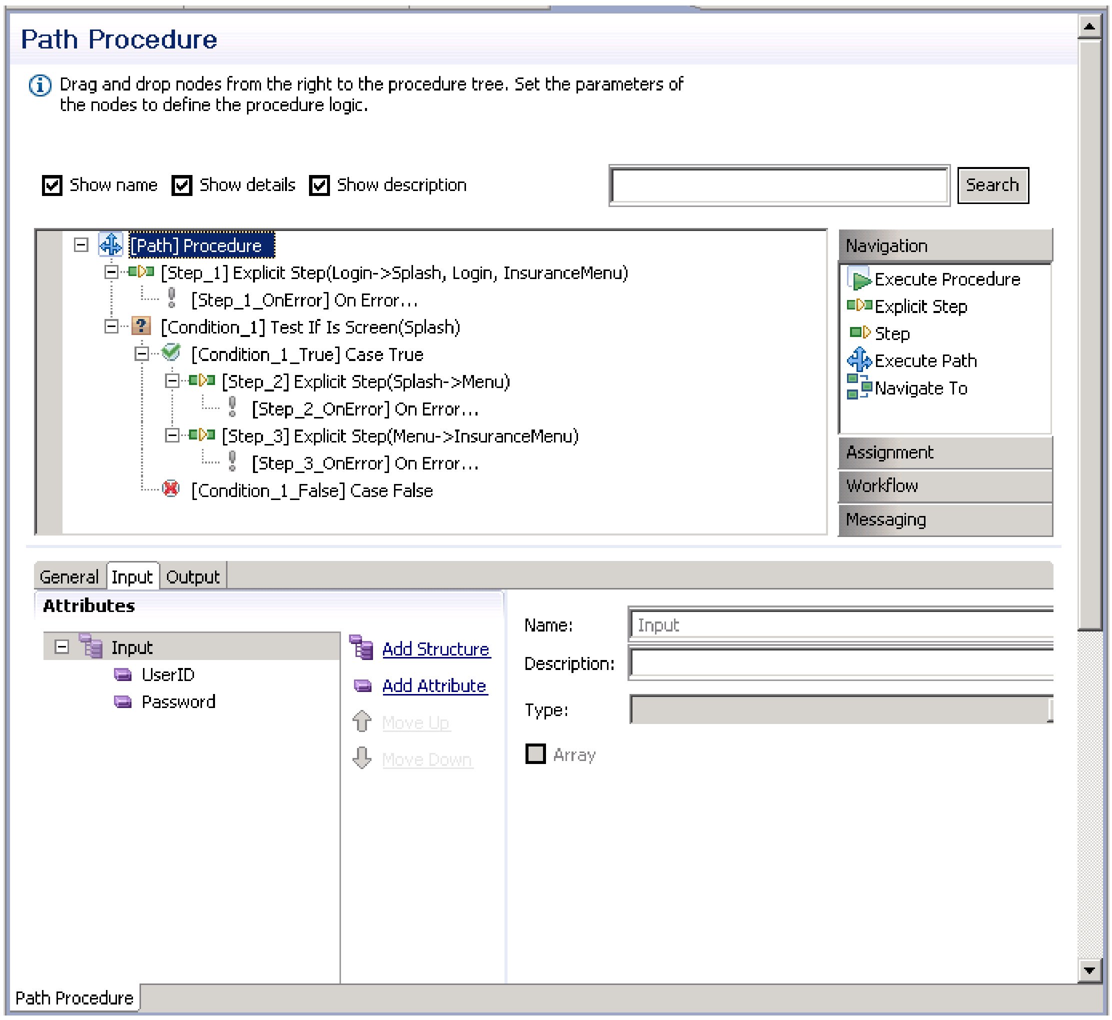
Includes a description of the node in the procedure details.

The procedure nodes are listed on the right, and can be placed within the procedure by dragging and dropping them to the relevant place within the procedure process. Clicking on one of the titles, such as Navigation, will display all the available nodes within navigation.

In the Procedure tree, right-clicking on a node will display the actions available for that node. These may include actions such as expanding or collapsing the node. The root node includes saving an image of the procedure to a file.

Path Procedures

Path Procedure Panel



Path Procedure

Drag and drop nodes from the right to the procedure tree. Set the parameters of the nodes to define the procedure logic.

☑ Show name ☑ Show details ☑ Show description

```
[Path] Procedure
  [Step_1] Explicit Step(Login->Splash, Login, InsuranceMenu)
    [Step_1_OnError] On Error...
  [Condition_1] Test If Is Screen(Splash)
    [Condition_1_True] Case True
      [Step_2] Explicit Step(Splash->Menu)
        [Step_2_OnError] On Error...
      [Step_3] Explicit Step(Menu->InsuranceMenu)
        [Step_3_OnError] On Error...
    [Condition_1_False] Case False
```

Input
- UserID
- Password

Name: Input

Description:

Type:

☐ Array

Show name
: Includes the name of the node in the procedure details.

Show details
: Includes details of the node in the procedure details.

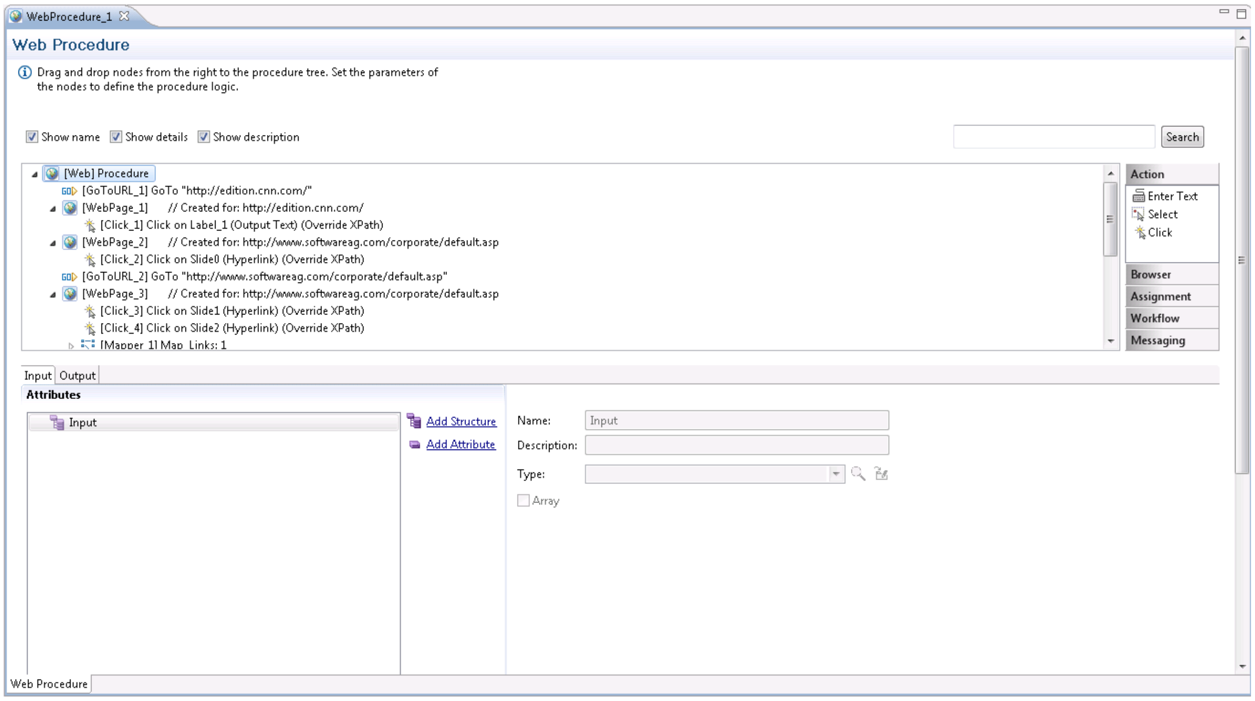
Show description
: Includes a description of the node in the procedure details.

The procedure nodes are listed on the right, and can be placed within the procedure by dragging and dropping them to the relevant place within the procedure process. Clicking on one of the titles, such as Navigation, will display all the available nodes within navigation.

In the Procedure tree, right-clicking on a node will display the actions available for that node. These may include actions such as expanding or collapsing the node. The root node includes saving an image of the procedure to a file.

Web Procedures

Web Procedure Panel



Drag and drop nodes from the right to the procedure tree. Set the parameters of the nodes to define the procedure logic.

☑ Show name ☑ Show details ☑ Show description

```
[Web] Procedure
    [GoToURL_1] GoTo "http://edition.cnn.com/"
    [WebPage_1]     // Created for: http://edition.cnn.com/
        [Click_1] Click on Label_1 (Output Text) (Override XPath)
    [WebPage_2]     // Created for: http://www.softwareag.com/corporate/default.asp
        [Click_2] Click on Slide0 (Hyperlink) (Override XPath)
    [GoToURL_2] GoTo "http://www.softwareag.com/corporate/default.asp"
    [WebPage_3]     // Created for: http://www.softwareag.com/corporate/default.asp
        [Click_3] Click on Slide1 (Hyperlink) (Override XPath)
        [Click_4] Click on Slide2 (Hyperlink) (Override XPath)
    [Mapper_1] Map  Links: 1
```

Input

Name: Input
Description:
Type:
☐ Array

Show name
: Includes the name of the node in the procedure details.

Show details
: Includes details of the node in the procedure details.

Show description
: Includes a description of the node in the procedure details.

The procedure nodes are listed on the right, and can be placed within the procedure by dragging and dropping them to the relevant place within the procedure process. Clicking on one of the titles, such as Navigation, will display all the available nodes within navigation.

In the Procedure tree, right-clicking on a node will display the actions available for that node. These may include actions such as expanding or collapsing the node. The root node includes saving an image of the procedure to a file.

Configuring the Proxy Settings

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:

```
<MainConfiguration>
        ...
        <ServerConfiguration>
        ...
           <webProcedureConfig>
              <proxyHostname></proxyHostname>
              <proxyPort></proxyPort>
              <proxyUsername></proxyUsername>
              <proxyPassword></proxyPassword>
           </webProcedureConfig>
        </ServerConfiguration>
</MainConfiguration>
```

What is an XPath?

ApplinX deciphers the Web page and finds elements using standard W3C technology called XPath. XPath, the XML Path Language, is a query language for selecting elements from an XML document. XPath was defined by the World Wide Web Consortium (W3C) and further details can be found at their site. When recording the Web procedure, and capturing an element, ApplinX finds the most suitable XPath that will locate the element in runtime. The XPath technology provides flexibility to locate a specific element using various XPaths. The format of an XPath can vary, for example, it can be of a format which only looks for a specific HTML attribute such as a name or ID within the Web page (recommended) or it can be a canonical XPath which provides the whole path, detailing the hierarchy of the tags which point to the element. If there is no name or ID, or if the name/ID is not generic enough, (i.e. if it is specific to a value that is likely to change in different instances of the page), alternative XPaths should be considered. The alternative suggestions by ApplinX are options which include various levels of hierarchy. The objective is to use an XPath which is based on a "solid" name/ID, with as few as possible levels of hierarchy as the fewer levels of hierarchy, the more stable the XPath is.

Handling Inline Frames

The suggested XPath has an additional level, and when there are inline frames, the XPath combines the web page and iframe's XPath into one XPath, to simplify the use within ApplinX.

For example:

The Web pages standard XPath (refers to the first iframe tag in the main document): `/html/body/iframe[1]`.

The IFrames standard XPath (refers to the second input tag under the first div tag, inside the iframe's document): `/html/body/div[1]/input[2]`.

The combined XPath used by ApplinX: `/html/body/iframe[1]/html/body/div[1]/input[2]`.

TroubleShooting

- What to do when the Web Procedure fails to run?
- Downgrading your Internet Explorer Version

What to do when the Web Procedure fails to run?

The Web Procedure can fail to run for a number of reasons such as an element cannot be found on the page or unexpected behavior when running JavaScripts. Some of these issues can be solved by adjusting parameters in the *<ApplinX installation>/config/gxconfig.xml* file. Following is a list of possible problems with a recommendation as to what to do and when relevant, which parameter to configure. Below is a snipet of the code as it appears within the gxconfig.xml file.

Timeout error: When running the Web Procedure, sometimes an error message is displayed indicating that a timeout occurred, and that the procedure failed and did not run successfully. This can be for a number of reasons:

- A specific element used in the procedure was not found on the page. In this case, you should try the following:
  1. The XPath defined may not capture the correct element. Run the procedure from within the Designer and follow the output in the Console area.
  2. Refine the XPath to make it more robust. Refer to **What is an XPath?**
  3. Change the ApplinX Server log level to "trace" to help you understand where the problem may lie.
  4. If you know that this element is on the page, you should try to change the `waitElementTimeout` parameter and see if after extending this time the element is found.
- Required resources are not loaded on the page within a certain amount of time. You can confirm that this is your problem by setting the ApplinX log level to "trace" and see that the actions/web element content retrieval is performed on the previous page. Change the `navigationTimeout` parameter to try and solve this.

Unexpected behavior when running JavaScript methods: If in a scenario which relies on the execution of a JavaScript method, the results are not as expected. Changing the `javascriptTimeout` may solve this.

CSS related problem: When there is a scenario that relies on the evaluation of CSS rules, and the results are not as expected, setting the `cssEnabled` parameter to enabled may solve this.

```
<MainConfiguration>
        ...
        <ServerConfiguration>
        ...
           <webProcedureConfig>
                <waitElementTimeout>30000</waitElementTimeout>
                <navigationTimeout>90000</navigationTimeout>
                <javascriptTimeout>30000</javascriptTimeout>
                <cssEnabled>false</cssEnabled>
           </webProcedureConfig>
        </ServerConfiguration>
</MainConfiguration>
```

Downgrading your Internet Explorer Version

The Web Integration solution requires recording the browser activity in Internet Explorer 7 or 8. If you have Internet Explorer 9, it is possible to downgrade to version 8 by removing the version 9 upgrade:

1. Close all programs.
2. Click **Start**, and then click **Control Panel**.
3. Click **Uninstall a Program** within the Programs category .
4. In the **Tasks** pane, click **View Installed Updates**.
5. In the list of installed updates, double-click *Windows Internet Explorer 9*.
6. Once Internet Explorer 9 has been uninstalled from your system, the computer will require a reboot. After rebooting, your computer should revert back to Internet Explorer 8.

General Nodes (Relevant for Flow, Path and Web Procedures)

Flow, Path and Web procedures may 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 are to be executed. Some nodes are only available when using Flow Procedures (refer to **Flow Procedure Nodes**), some nodes are only available when using Path Procedures (refer to **Path Procedure Nodes**), and some nodes are only available when using Web Procedures (refer to **Web Procedure Nodes**).

Note: Tree nodes have nested scopes, much like blocks in programming languages. Objects and values are available for mapping in the node in which they are defined and their child nodes, but not in the parent nodes.

- Execute Procedure Node
- New Object Node

- Create Mappings Node
- Merge Arrays Node
- Throw Exception Node
- Log Message Node
- Send Mail Node
- Loop While Node
- For Each Node
- Test If Node
- Switch Node
- Try/Catch Node
- Exit Node
- Sleep Node

Execute Procedure Node

Used in procedures to execute a procedure and return the procedure's output. It contains a Mapper and defines Procedure Input, information about the session (for Path procedures and Program procedures) and Connection Properties (Program Procedures).

To Create an Execute Procedure Node

1 In the **Procedure** editor, drag and drop the **Execute Procedure** node (from within the Navigation divider) to the relevant position within the procedure.

2 In the bottom half of the screen, select the procedure.

3 Map necessary inputs from the source to the target. The target includes inputs, information about the session (for Path procedures and Program procedures) and Connection Properties (Program Procedures) such as user ID and password).

4 After the node is executed, the Procedure Output element, depending on the selected Procedure Output definition, is available at the procedure scope.

New Object Node

In the New Object node define temporary data structure which can be used within the context. Using the mapping tool, you can map values to parameters within the node, and later use the mapper to access the values of these parameters in other nodes within the context.

To create a New Object node

1 In the Procedure editor, drag and drop the New Object node (from within the Assignment divider) to the relevant position within the procedure.

2 The object can be defined as simple attribute, array, simple structure or array of structures. Click Add attribute or Add Structure as required. Enter a name and determine whether it is an array. It is also possible to provide a default value.

Create Mappings Node

This node enables you to create mappings between scope objects/variables or expression values to any output or object defined in the flow procedure's scope. Refer to using the mapper.

Merge Arrays Node

The Merge Array node is used to create an array of structured elements (objects) from a number of arrays of simple elements of the same size (such as inputs, or outputs from another procedure etc.). The iteration runs on one of the simple elements and therefore if the size of the arrays of all the simple elements is not the same, the results of the procedure may be lacking.

The following example will guide you through a basic example of the use of the Merge Arrays node in flow procedures. The source data is taken from three string array inputs (Name, Age and Address). The data received as a result of running the procedure will be placed in an array of a structure (in this case, a Business Entity (named Person), which includes Name, Age and Address attributes).

1. Create a Data Structure entity. In the Name field type Person and save the entity.
2. Open the entity in the Editor and add three attributes: Name, Age and Address (refer to Business Entity and Creating Business Entity Attributes). Save the Data Structure.
3. Create a flow procedure and add three array inputs: Name, Age and Address to the Flow Procedure node (refer to Defining Inputs and Outputs). Add an output called People. In this Output, add a structure. In the Type field select the data structure Person and select the Array check box.
4. In the Procedure editor, drag and drop the Merge Arrays node (from within the Assignment divider) to the relevant position within the procedure.
5. Click Select to select the object, where the data received as a result of running the procedure will be placed. The Modify Expression dialog box is displayed.
6. Double-click People to determine that the data will be mapped to the output called People. Click OK.
7. Click For Each Select Object to determine the input array that the procedure will run over. The Modify Expression dialog box is displayed.
8. Double-click In\Name to determine that the procedure will run over the array of names. Click OK.
9. In the bottom panel, expand In to view the list of inputs.
10. Click and drag In\Name to People\Name, creating a line between the two. In the same way, create a line between In\Age to Person\Age and between In\Address to Person\Address. When the procedure is run, an array of structures will be created (People).
11. Click Condition to filter some of the mappings. For example, to avoid mapping values from empty entities of the source arrays.

12. Click the Play icon in the debugger toolbar to test the procedure. Enter values for the input strings for three names, ages and addresses.
13. Click OK. The bottom panel displays the procedure inputs, outputs and status. The In node displays the string array data entered in the Get Flow Input dialog. The Out node displays structures of the Business Entity Person, which consists of the name, age and address of the string arrays, for all indexes in the array.

Throw Exception Node

Throws an exception with the message specified by the expression.

Log Message Node

Writes a log message specified by the expression. The expression also includes the level of the message: NORMAL, WARNING or ERROR.

Send Mail Node

The Send Mail node sends mail to the designated email addresses. Fill in the To and CC fields as well as a subject for the mail. In the From field enter the address which you would like to appear in the From field on the mail. When this field is left empty, the Default From address defined in the Server Configuration will be used.

Loop While Node

Executes its child nodes while a specified condition is true.

For Each Node

Executes the child nodes for each item of the specified array that matches the condition.

Test If Node

An "If" expression: if result is true executes its Case True (container) child node, otherwise the Case False (container) child node is executed.

Switch Node

A Switch Screen function, depending on the switch value (the current screen), selects one of several possible cases. Each case is defined to handle one or more screens. If the current screen isn't one of the screens specifically handled in one of the cases then the default case is used. There can't be more than one case that handles a specific screen.

To create a Switch Node

1. In the Procedure editor, drag and drop the Switch node (from within the Workflow divider) to the relevant position within the procedure.
2. Click on <expr> at the bottom of the screen, to define the value of the Switch function.
3. To add additional cases, right-click on the Switch node and select Add Child>Case. In the Case Values panel, click on the empty row and enter the value. Press ENTER to confirm the value. It is possible to add multiple values to a single case node enabling the same case functionality to be used for a number of values.

Try/Catch Node

The Try node contains the sections of nodes that might potentially throw exceptions and the Catch node contains nodes that handle exceptions.

To create a Try/Catch Block Node

1. In the Procedure editor, drag and drop the Try/Catch node (from within the Workflow divider) to the relevant position within the procedure.
2. Right-click Try and define the nodes that are to be part of the Try Block and may potentially throw exceptions.
3. Right-click Catch and define the nodes that will handle the exceptions.

Exit Node

The Exit node is used to exit the procedure.

Sleep Node

The Sleep node causes the procedure to pause for the defined number of milliseconds.

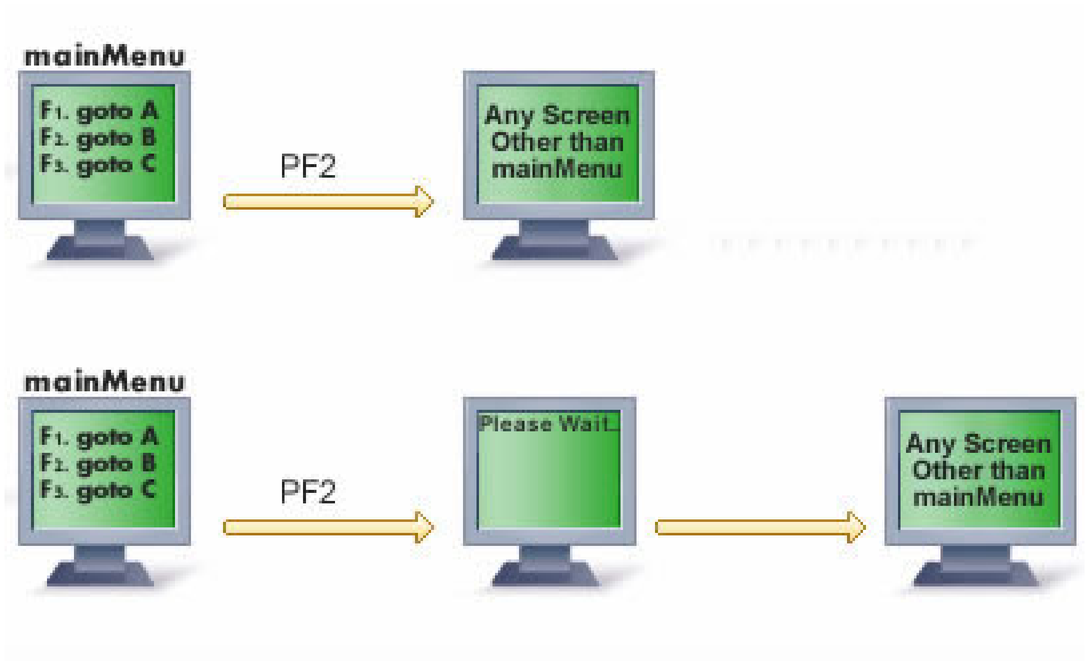
To create a Sleep Node

1. In the Procedure editor, drag and drop the Sleep node (from within the Workflow divider) to the relevant position within the procedure.
2. Click on the expression to determine the number of milliseconds the procedure should "sleep".

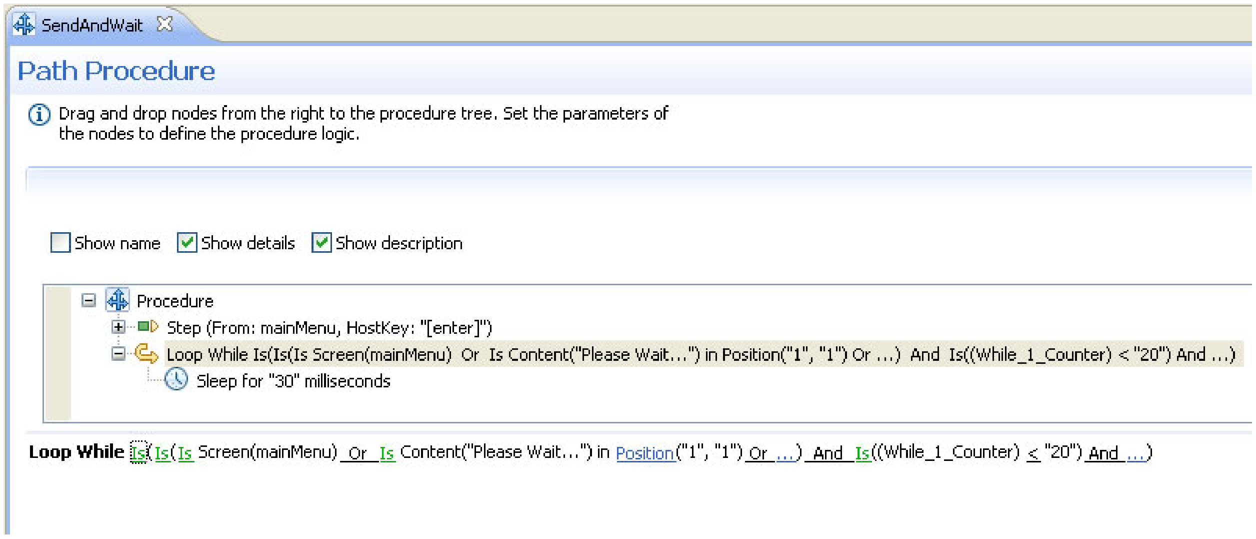
This node is typically used to create complicated customized wait conditions, where regular wait conditions cannot cover the complexity of the host behavior.

Implementation

Pressing PF2 when in a screen named "mainMenu" should take us to another screen right away, however, every now and then the host displays a blank screen with a "please wait" message at the top and only after a second or so displays the desired screen.



This example enables creating a Wait condition that pauses the procedure as long as the current screen is still "mainMenu" or as long as the "Please wait" message appears at the top of the screen. In addition it also prevents the procedure from hanging in an infinite loop from iterating more than 20 times. To accomplish this, a "Loop While" node with a complex condition, placing a Sleep Node inside the loop, should be used.



Drag and drop nodes from the right to the procedure tree. Set the parameters of the nodes to define the procedure logic.

☐ Show name ✓ Show details ✓ Show description

```
Procedure
    Step (From: mainMenu, HostKey: "[enter]")
    Loop While Is(Is(Is Screen(mainMenu)  Or  Is Content("Please Wait...") in Position("1", "1") Or ...)  And  Is((While_1_Counter) < "20") And ...)
        Sleep for "30" milliseconds
```

Loop While Is(Is(Is Screen(mainMenu) Or Is Content("Please Wait...") in Position("1", "1") Or ...) And Is((While_1_Counter) < "20") And ...)

Flow Procedure Nodes

- Create Emulation Session Node
- End Session Node
- Create DB Session Node
- DbSelect Node
- DbExecute Node
- Rollback Node
- Commit Node
- Parallel Actions Node

Create Emulation Session Node

Used in procedures to create and initiate a new host session.

To Create a Create Emulation Session Node

1 In the Procedure editor, drag and drop the Create Emulation Session node (from within the Session divider) to the relevant position within the procedure.

2 Select a connection pool from the list in order to create a new session to work with a connection of the selected connection pool. If you do not select a connection pool, a new session will be created against the host.

3 Refer to the Mapper for details on mapping.

Note: The Host user name and Host password fields enable you to establish an SSH connection. The Device name enables you to provide the host with the relevant device name.

4 Once the new session is created, its session ID and device are available as outputs of the Create Emulation Session node.

End Session Node

Used in procedures to end an existing ApplinX Host, RPC or Database session. To create an End Session Node: in the Procedure editor, drag and drop the End Session node (from within the Session divider) to the relevant position within the procedure. Refer to the Mapper for details on mapping.

Note: In the case of connection pools (RPC or emulation session), “end session” does not mean closing the session. The connection returns to the pool, and further action depends on how the pool has been configured. See **Connection Pool** in this section and *Defining ApplinX RPC Application Parameters* under *Developing an ApplinX Application*.

Create DB Session Node

This node is used in procedures to create and initiate a database session against a database in order to perform a transaction. A transaction is a number of statements, which together create a process against the database. When performing one or a number of separate, unconnected statements, use DbSelect and DbExecute, which open and close sessions independently.

To Create a DB Session Node

1 In the Procedure editor, drag and drop the Create Emulation DB Session node (from within the Database divider) to the relevant position within the procedure.

2 In the attributes area (the bottom half of the screen), select the database.

3 Select **Automatically commit** to automatically commit every execute statement as it is completed (in such a case it is not possible to rollback). If **Automatically commit** is not selected, you must use the commit node at the end of the transaction in order for the execute statements be saved to the database.

DbSelect Node

Executes an SQL statement that returns a single ResultSet object. Refer to DbExecute.

To create a DbSelect node

1 In the Procedure editor, drag and drop the DbSelect node (from within the Database divider) to the relevant position within the procedure.

2 In the attributes area (the bottom half of the screen), select the database.

3 In the **SQL statement** text box, write the statement that you want to execute. If the statement has parameters that should be provided at runtime, use the following syntax:

- `$(varName)` for the variable you want to convert to the correct SQL format during the runtime (add an apostrophe before and after the variable).
- `$(!varName)` for the variable that will be used "as is" (use it for table or field names).

You will see the defined variables in the right panel of the mapper. You can map the elements from the left panel to those variables.

4 In the **Output parameter**'s panel, define the fields returned by the SQL statement (the order is important).

Use the refresh button to import the output parameters directly from the database (be sure the SQL statement is valid). After the node is executed the DbSelect output element, depending on the defined output parameters, is available at the procedure scope.

DbExecute Node

Executes an SQL INSERT, UPDATE or DELETE statement.

To create a DbExecute node

1 In the Procedure editor, drag and drop the DbExecute node (from within the Database divider) to the relevant position within the procedure.

2 From the database combo box, select the ApplinX database entity.

3 In the SQL statement text box, write the statement you want to execute. If the statement has parameters that should be provided at runtime, use the following syntax:

- `$(varName)` for the variable you want to convert to the correct SQL format during the runtime (add an apostrophe before and after the variable).
- `$(!varName)` for the variable that will be used "as is" (use it for table or field names).

You will see the defined variables in the right panel of the mapper. You can map the elements from the left panel to those variables.

The DbExecute RecordCount element contains either the row count for INSERT, UPDATE or DELETE statements, or "0" for SQL statements that return nothing.

Rollback Node

Drops all changes made since the previous commit/rollback and releases any database locks currently held by the database session.

To create a Rollback node

1 In the Procedure editor, drag and drop the Rollback node (from within the Database divider) to the relevant position within the procedure.

2 The existing database session ID must be provided.

Commit Node

Ensures all changes made since the previous commit/rollback are permanent and releases any database locks currently held by the database session. The existing database session ID should be provided.

Parallel Actions Node

The Parallel Actions node will perform actions in parallel. The Parallel Actions node enables these actions to be performed at the same time rather than one after the other. Only certain types of nodes can be used as Parallel Actions:

- Send Mail
- Execute Procedure
- Create Emulation Session
- Create DB Session
- DbSelect
- DbExecute
- Rollback
- Commit
- Create RPC Session
- End Session

To create a Parallel Actions Node

1 In the Procedure editor, drag and drop the Parallel Actions node (from within the Workflow divider) to the relevant position within the procedure.

2 Right-click on the node and select Add Child and then the relevant node.

Note: Failure of one action in the node will cause the node, together with all other actions, to fail. The error returned is always that of the first child node that failed.

Path Procedure Nodes

Explicit Step Node

Step is the basic building block of a path. It defines a single act of navigation between a source screen\screen group and a target screen\screen group in the host. A step defines the data to set into fields of the source screen\screen group and the key to send ([Enter], [PF3]...). It may define "wait" expressions, to make sure the host has enough time to process the input and reach the target screen\screen group. Once the target screen\screen group is reached, it may be collected (for path response) or map values from its fields to the Procedure's output structures.

To Create an Explicit Step Node

1 In the Procedure editor, drag and drop the Step node (from within the Navigation divider) to the relevant position within the procedure.

2 Select the Source screen\screen group: The Source screen\screen group selection box allows the selection of the start screen\screen group of the step. This screen\screen group is used for validation at the beginning of the step, and to build the screen's schema for the input tab. Multiple Source screens\screen groups can be defined.

The Target screen\screen group selection box allows the selection of the end screen or screens of the step. This screen\screen group is used for validation after sending data to the host, for execution of "wait for screen" expressions and to build the screen's schema for the output tab. Multiple Target screens can be defined.

3 The input tab uses the mapper component to map values into the source screen(s), to set the cursor, and to send host keys:

Source screens - The mapping target shows the selected source screens and their mapped fields. If the screen is unknown, no fields are available.

Cursor - The mapping target will show a Cursor node with row, column and field. Mapping is possible to either row and column or field. In order to position the cursor in a certain application field, map a string containing the field's name to the cursor. It is also possible to set the cursor position using multiple fields. Instead of just mapping the field name to the field part of the cursor schema, the string contains the required field index, for example:

"action[3]" or "action[$(index)]" where the token $(index) is mapped to another runtime value.

Send host keys - The mapping target shows a HostKeys node and default keys expression ([ENTER]). The keys expression is based on the Free Text expression, but it allows adding the key strings by selecting them from a predefined list.

4 Define the Wait conditions in the Wait tab: Wait conditions indicate to the ApplinX Server that a screen has fully arrived. A Wait condition is the condition by which the ApplinX Server decides that the host has finished sending screen data. The screen is then returned to the ApplinX Base Object. It uses the expression wizard mechanism to build wait conditions, whereby multiple conditions can be combined. Processing continues when all conditions are successfully met.

Click on (<wait>) to define the wait expression. To define additional wait conditions, click on <...>.

Note: When no wait conditions are defined, there is a default timeout of 10 seconds for each step.

Character-mode hosts (for example UNIX or VT hosts): As character-mode hosts are character stream based and never stop sending data, it is necessary to divide the data sent to screens, by defining Wait conditions.

5 Define outputs: The output tab contains a Screen Mapper node for each Target screen. The mapper area displays the output schema of the screen that is selected in the "Map from target screen" selection box. On the other side it displays the scope elements and input schema, allowing mapping values from the current screen, to other structures. When the Path procedure is executed using the ABO, the entire screen can be added to the path response using the "Send to Base Object" checkbox.

6 Enter the repeat limit: The same step may repeat itself several times. In order to avoid infinite loops it is possible to set a limit to the number of times the step may be repeated during one execution of the path. It is recommended to set this limitation using the flow logic, though it can also be done by setting a repeat limit.

Step Node

This node defines the data to set into the source screen and the key to send. It may define "wait" expressions, to make sure the host has enough time to process the input.

To create a Step node

1. In the Procedure editor, drag and drop the Step node (from within the Navigation divider) to the relevant position within the procedure.
2. Select the Source screen: The Source screen selection box allows the selection of the start screen of the step. This screen is used for validation at the beginning of the step, and to build the screen's schema for the input tab. Multiple Source screens can be defined.
3. Define the content of the current screen: This tab contains a Screen Mapper node for each Target screen. The mapper area displays the output schema of the screen that is selected in the "Source" selection box. On the other side it displays the scope elements and input schema, allowing mapping values from the current screen, to other structures. When the Path procedure is executed using the ABO, the entire screen can be added to the path response using the "Send to Base Object" checkbox.
4. The input tab uses the mapper component to map values into the source screen(s), to set the cursor, and to send host keys:

   Source screens - The mapping target shows the selected source screens and their mapped fields. If the screen is unknown, no fields are available.

   Cursor - The mapping target will show a Cursor node with row, column and field. Mapping is possible to either row and column or field. In order to position the cursor in a certain application field, map a string containing the field's name to the cursor. It is also possible to set the cursor position using multiple fields. Instead of just mapping the field name to the field part of the cursor schema, the string contains the required field index, for example:

   "action[3]" or "action[$(index)]" where the token $(index) is mapped to another runtime value.

   Send host keys - The mapping target shows a HostKeys node and default keys expression ([ENTER]). The keys expression is based on the Free Text expression, but it allows adding the key strings by selecting them from a predefined list.
5. Define the Wait conditions in the Wait tab: Wait conditions indicate to the ApplinX Server that a screen has fully arrived. A Wait condition is the condition by which the ApplinX Server decides that the host has finished sending screen data. The screen is then returned to the ApplinX Base Object. It uses the expression wizard mechanism to build wait conditions, whereby multiple conditions can be combined. Processing continues when all conditions are successfully met.

   Click on (<wait>) to define the wait expression. To define additional wait conditions, click on <...>.

Note: When no wait conditions are defined, there is a default timeout of 10 seconds for each step.

Character-mode hosts (for example UNIX or VT hosts): As character-mode hosts are character stream based and never stop sending data, it is necessary to divide the data sent to screens, by defining Wait conditions.

6 Enter the repeat limit: The same step may repeat itself several times. In order to avoid infinite loops it is possible to set a limit to the number of times the step may be repeated during one execution of the path. It is recommended to set this limitation using the flow logic, though it can also be done by setting a repeat limit.

Execute Path Node

This node calls another Path procedure or map (this is equivalent to the "Inner path" steps of the Path entity). Just as in the Step node, you are required to define the source and target screens\screen groups. The mapper area shows the selected path's input schema (application fields and variables).

Nodes that are placed after the Path node in the same scope can access the path execution output schema using mappers.

To create a Path node

1 In the Procedure editor, drag and drop the Path node (from within the Navigation divider) to the relevant position within the procedure.

2 In the Path field, select the Path procedure to execute.

3 Select the source and target screens\screen groups.

4 Map values to the fields and variables in the path.

5 Properties Tab (optional) • Enter the repeat limit: The same step may repeat itself several times. In order to avoid infinite loops it is possible to set a limit to the number of times the step may be repeated during one execution of the path. It is recommended to set this limitation using the flow logic, though it can also be done by setting a repeat limit.

Navigate To Node

Use this node to use the Application Map to navigate to a specific screen.

To create a Navigate To node

1 In the Procedure editor, drag and drop the Navigate To node (from within the Navigation divider) to the relevant position within the procedure.

2 Select the required screen from the list of screens.

Screen Mapper Node

When the current screen matches the screen defined in the Screen Mapper node, the defined mappings are performed, allowing retrieving values from the host screen schema. The mapper area displays the output schema of the screen that is selected in the Map from current screen selection box. On the other side it displays the scope elements and the Path procedure's input schema, allowing mapping values from the current screen, to other structures. When the Path procedure is executed using the ApplinX Base Object, the entire screen can be added to the path response using the Send to Base Object checkbox. The screen schema contains the following structures:

- Cursor - the cursor structure allows reading the current screen's cursor position (row, column). In addition, if the cursor is in an application field, its name can be retrieved too.
- Application Fields - the application fields which are mapped to the screen are visible in the screen's schema, and their content can be retrieved using the mapper. For other field attributes, use expressions.
- Host Table - when the host screen contains a host table, it is reflected in the screen schema as an array of row objects, whose simple attributes are the table's columns. The application fields that represent the columns will also appear as separate arrays outside the table's schema, providing more flexibility when mapping.

Switch Screen Node

A Switch Screen function, depending on the switch value (the current screen), selects one of several possible cases. Each case is defined to handle one or more screens. If the current screen isn't one of the screens specifically handled in one of the cases then the default case is used. There can't be more than one case that handles a specific screen.

To create a Switch Screen node

1. In the Procedure editor, drag and drop the Switch Screen node (from within the Workflow divider) to the relevant position within the procedure.
2. To add cases, right-click on the Switch... node and select Add Child>Switch Screen Case. In the Case panel, click on Assign Screens to view a list of the screens. It is possible to select and add multiple screens.

Web Procedure Nodes

GoTo URL Node

Used in the Web Procedure to navigate to a URL. This node reflects the URL entered in the Web Procedure Recorder. In the Editor, you can add navigation to other Web pages using this node (from within the Browser tab to the right of the procedure). It contains the URL and can be edited by right-clicking on the URL link and selecting **Open**. The URL passes parameters to the Web page. For example, in the following URL, the parameter `country` receives the value "US": *ht-tp://www.foo.com/bar?country=us*. Within the procedure Editor, you can edit the URL as as any **Free Text** expression and replace the value with a token: *http://www.foo.com/bar?country=$(country)*. This allows replacing the token dynamically, for example, using a Procedure Input attribute to set the value.

Web Page Node

For every page where an action was performed, a new Web Page node is created. This node includes all the actions performed on the page as child nodes. The child nodes of the Web Page can use all the elements on this page. When selecting the Web Page Node, you can see all the Web Elements defined in it. It is possible to add more elements to a Web Page by right-clicking on the root node of the Web Page. For each Web Element, you can copy its XPath or delete it.

Enter Text Node

This action node enables entering text within a specific input element within a Web Page. In the Editor you can add additional Enter Text nodes or edit existing ones. Note that when the action relates to an element within a list of elements, you must define an expression that specifies on which index within the list the action should be performed.

Value: The value that will be entered in this element when running the Web procedure. Click on the value link and use expressions to define the value.

Web Element: Select a web element from the list of elements of this type that have been captured in this Web Page. When editing an existing Enter Text node, you can use the override XPath option to change the element.

Select Node

This action node enables selecting a specific value that will be placed in the element of a Web Page when running the procedure. In the Editor you can add additional Select nodes or edit existing ones. Note that when the action relates to an element within a list of elements, you must define an expression that specifies on which index within the list the action should be performed.

Element Type: It is possible to toggle between the element types: Check Box, Drop-Down List and Radio Buttons and change the type.

Web Element: Select a web element from the list of elements of this type that have been captured in this Web Page. When editing an existing Enter Text node, you can use the override XPath option to change the element.

Value: You can determine the value to select in this element. For drop-down lists it is possible to determine that these values will be according to according to the value, according to the index or according to name. For Radio buttons, it is possible to determine that these values will be according to index or value.

For example (in a drop-down list of countries list) :

by name: "United States" - this is what the user see in the drop-down list options

by value: "USA" - this is the value that the drop-down list will send in the form to the web server

by index: 5 - the country United States was the 5th option in the drop-down list

Click Node

This action simulates clicking on an element such as a button or hyperlink in a Web Page, when running the procedure. Note that when the action relates to an element within a list of elements, you must define an expression that specifies on which index within the list the action should be performed.

Web Element: Select a web element from the list of elements of this type that have been captured in this Web Page. Once a Web Element is selected, the type of element (such as hyperlink), and the Override XPath option enabling editing the XPath, also appear in the format.

General Expressions (relevant for Flow, Path and Web Procedures)

Expression Types

Expressions are used in nodes and child nodes of flow procedures to compute and assign values to variables and to help control the execution flow of a procedure. The object of an expression is to perform the computation indicated by the elements of the expression and to return a value that is the result of the computation. The expression types available vary according to the node you are defining.

- EmptyString, TRUE and FALSE Expressions
- Free Text
- Value Of
- Count Of
- Conditional Operator
- String Array
- Execute Procedure
- Now
- Create Date
- To Date
- Date Part
- Compare
- Logical And/Or
- Is Null
- Calculate
- Ceil
- Floor
- Round
- Absolute
- Concat
- Trim
- StrIn
- SubString
- Replace String
- Change Case
- StringLength
- Reverse
- FormatDate
- Format Number
- Extract Number

- Character

EmptyString, TRUE and FALSE Expressions

Standard fixed ApplinX syntax, used for these functions.

Free Text

In the **Free Text** dialog box, type in any text and add tokens in order to use values from the context. Click Finish. Click on the token link to define the variable. When previewing the text, line breaks are replaced with semicolons to simplify the display.

Example

Enter text and replaceable tokens in the **Free Text** tab: "Your account number is - $(var)"." $(var)" being a replaceable token. Click **Finish**. Use expressions to define a value for the token.

Refer to Mapper

Value Of

The **Value Of** expression returns the value of the selected object.

To define the Value Of expression

1 Select an item from the available scope.

2 Double-click to select this item as the expression's value.

Note: When selecting an expression which has a complex input or output structure which includes arrays, it is possible to select a specific index.

Count Of

The **Count Of** expression returns a count of an array item.

To define the Count Of expression

1 Select an array item.

2 Double-click to select this item as the expression's value.

Conditional Operator

The **Conditional Operator** expression is short-hand for an if-else statement. The Conditional Operator returns `<expr1>` if `<condition>` is true or returns `<expr2>` if `<condition>` is false.

Format

```
If <condition> Then <expr1> Else <expr2>
```

Implementation

Click on each `<expr>` and define the expression.

Example

```
If (( In/AccountNumber ) = 23453) Then TRUE Else FALSE
```

String Array

Returns a string array.

Format

```
StringArray (expr, ...)
```

Implementation

Use the `<expr>` link to define the first string. To define additional strings click the "..." link.

Execute Procedure

An expression that executes an ApplinX Procedure and returns its output.

Implementation

In the Input tab, select a procedure. Map values to the procedure's input using the mapper. In the Output tab, click the expression's output.

Now

Now expression returns the current date and time according to the setting of your computer system's date and time.

Create Date

Create Date returns a date for a specified year, month, day, hour, minutes and seconds.

Format

```
CreateDate (<year> , <month> , <day> , <hour> , <minute> , <second>)
```

Implementation

Click the links to define the expressions for the different parts of the date/time.

Example

`CreateDate (1982 , 07 , 19 , 09 , 20 , 13)` will return "1982-07-19 09:20:13:000"

To Date

To Date creates a date from a date/time string according to the given date format.

Format

```
ToDate (<datestring> , <format>)
```

Implementation

Click the links to define the expressions for the date string and format.

Example

`ToDate (19/07/1982 , dd/MM/yyyy)` will return" 1982-07-19 00:00:00:000"

Date Part

Date Part extracts a part of the date (year, month, hour etc.) from a date expression.

Format

```
YearOf(<date>)

MonthOf (<date>)

DayOf(<date>)

HourOf(<date>)

MinuteOf(<date>)

SecondOf(<date>)
```

Implementation

Select the date part: year, month, day, hour, minute or second. Click on the date expression and define the Date expression.

Example

`YearOf (Now)` will return "2004"

Compare

Compare expression compares the values of two numeric or textual expressions.

Note: When comparing two null expressions, the function will return "false".

Format

```
Is (<expr> = <expr>)

Is (<expr> > <expr>)

Is (<expr> < <expr>)
Is (<expr> >= <expr>)

Is (<expr> <= <expr>)
```

Implementation

Click `'Is'/'Is not'` to switch between the two options. Select the required comparison operator. Click the `<expr>` links to edit.

Logical And/Or

An expression that applies a logical AND or OR to several boolean expressions.

Format

```
Is (<expr> AND ...)

Is Not (<expr> AND ...)
```

Implementation

Click `'Is'/'Is not'` to switch between the two options. Click the `<expr>` or `"..."` links to add expressions. Select the required boolean operator (AND or OR).

Is Null

Is Null checks whether the selected object does not have an actual during runtime.

To define the Is Null expression:

1 Select an item from the available scope.

2 Click to select this item as the expression's value.

Calculate

Calculate returns a calculation and may include variables and arithmetic calculations.

Implementation

Type in the calculation formula using digits and operators. Click **Finish**. Click on the variable link to define an expression. A token representing this expression will appear in the calculation.

Ceil

Returns the smallest value that is not less than the argument and is equal to a mathematical integer. The value is displayed in double format.

Format

`Ceil(<expr>)`

Implementation

Click `<expr>` to define the relevant expression.

Example

`ceil(2.645);` will return "3.0"

Floor

Returns the largest value that is not greater than the argument and is equal to a mathematical integer. The value is displayed in double format.

Format

`Floor(<expr>)`

Implementation

Click `<expr>` to define the relevant expression.

Example

`floor(2.645);` will return "2.0".

Round

Returns the closest integer to the argument.

Format

`Round(<expr>)`

Implementation

Click `<expr>` to define the relevant expression.

Example

```
round(2.500); will return 3.

round(2.499); will return 2.
```

Absolute

Returns the absolute value of the argument. The value is displayed in double format.

Format

`Absolute(<expr>)`

Implementation

Click `<expr>` to define the relevant expression.

Example

`Absolute(2.300); will return 2.0.`

Concat

Returns a string value containing the concatenation of two or more supplied strings.

Format

`Concat("<expr>",...)`

Implementation

Use the <expr> link to define the first string. To define additional strings click the "..." link.

Example

`Concat("John", "Smith",...)` will return "`JohnSmith`".

Trim

Trim expression returns a string containing a copy of a specified string with no leading or trailing spaces.

Format

`Trim(<expr>)`

Implementation

Click the `<expr>` to define the string expression to trim.

Example

`Trim(" John ")` will return "`John`".

StrIn

StrIn expression returns the position of the first occurrence of one string within another.

Format

```
StrIn (<string> , <substring>, <case sensitive>)

StrIn (<string> , <substring>, <case insensitive>)
```

Implementation

Click `<string> , <substring>` to define the string in which to search and the string to search for. The expression will search for the first occurrence of the second string within the first string. Toggle between **case insensitive** and **case sensitive** to determine case sensitivity.

Example

`StrIn ("Catwalk", "Cat")` will return "0"

`StrIn ("John", "Smith")` will return" -1 "

`StrIn ("Caterpillar", "pillar")` will return "6"

SubString

SubString expression returns a substring that begins at a specified location, and has a specified length.

Format

```
SubString (<string> , <start> , <length>)
```

Implementation

Click the links to define the original string, the start index and the required length of the sub-string.

Example

`SubString ("Caterpillar", 6, 6)` will return "pillar".

Replace String

Replaces either the first substring or all substrings in this string that match the given pattern, with the defined replacement.

Format

```
ReplaceString( <string> , <patternToReplace> , <replacement>, <ReplaceFirst> )
```

Implementation

Click the links to define the original string, the regular expression pattern to be replaced, the replacement string, and whether to replace just the first substring or the whole string.

Example

`ReplaceString("elephant", "e..a", "ega")` will yield the string "elegant".

Change Case

Change Case expression returns a string that has been converted to a specified case (lowercase or uppercase).

Format

`ToLowerCase (<expr>)`

`ToUpperCase (<expr>)`

Implementation

Select the relevant option to transform the expression to upper or lower case. Use the link to define the expression.

Example

`ToLowerCase ("JOHN")` will return "john" .

`ToUpperCase ("john")` will return "JOHN".

StringLength

StringLength expression returns the length of a string.

Format

`StrLen (<expr>)`

Implementation

Click the `<expr>` to define the string.

Example

`StrLen ("John")` will return "4".

Reverse

Reverse expression returns the reverse of a string expression.

Format

`Reverse (<expr> )`

Implementation

Click the `<expr>` to define the string expression.

Example

`Reverse ("caterpillar")` will return "rallipretac".

FormatDate

FormatDate expression converts a date/time object into a date/time string, according to the given date format.

Format

`FormatDate (<date> , <format>)`

Implementation

Click the links to define the expressions for the date/time object and format.

Example

`FormatDate (Now, "dd/MM/yyyy")`, Now expression being the current date and time, will return "19/07/1982".

Format Number

Formats a number according to the given format number.

Format

`FormatNumber( <number> , <format> )`

Implementation

Click the links to define the expressions for the number and format.

Example

For example if the number 18734573.07 is required as 18,734,573.07, use the format "#,##0.00". Refer to Number Format for further explanation about the format syntax.

Extract Number

Extract Number expression extracts a numeric value from a textual source number. When there is more than one number, it extracts the first number it locates. This expression may be used when needing to perform calculations on the source number.

Format

`ExtractNumber (<expr>, Decimal:dot)`

Implementation

Select the relevant decimal symbol: dot or comma. The separator that you do not select will be recognized as the thousand separator and will be removed. Click the link to define the source expression.

Example

When selecting the dot separator, `ExtractNumber ("1,000,876.321")` will return "1000876.321"
When selecting the comma separator, `ExtractNumber ("1.000.876,321")` will return "1000876.321"

Character

Character expression returns an ASCII or Unicode Character according to the decimal representation.

Implementation

Insert the character's ASCII code or Unicode value.

Example

Enter "13", the text in the **Value** field will display "carriage return" indicating the functionality.

Path Procedure Specific Expressions

Path Procedure Expressions

- Wait Expression
- Test Screen Name
- Test Field Attribute
- Test Screen Content
- Is Member of Group
- Host Keys
- Position
- Screen Name
- Field Attributes
- Screen Properties
- Screen Buffer
- Field Content
- Find Field Index
- Field Occurrences

Wait Expression

The Wait expression is accessed from the Wait tab in the Step node of the Path procedure. The Wait tab lists expressions that represent the step waits. A Wait condition is the condition by which the ApplinX Server decides that the host has finished sending screen data, and returns the screen to the ApplinX Base Object. It uses the expression wizard mechanism to build wait conditions, whereby multiple conditions can be combined. Processing continues when all conditions are successfully met.

Select the Wait type and update the relevant attributes by clicking the expression links. You can use the session viewer for easy definition of strings and positions.

Wait type: For Host quiet

Waits for the host to stop sending data.

Format

Wait for host quiet (<timeout> , <flicker>)

Implementation

Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: For String

Waits for a specific string to be written in a specific place on the screen.

Format

Wait for string (<string> , <startPosition> , <endPosition> , <timeout> , <flicker>)

Implementation

Select the Case sensitive checkbox to determine that the specific string is case sensitive. Click <string> to define the specific string. Click <startPosition> and <endPosition> to define the start position and end position that determine the boundaries of a rectangle inside which it is possible for the string to appear. The start position is mandatory, and the end position is optional. Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: For cursor

Waits for the cursor to reach a certain position on the screen.

Format

Wait for cursor (<startPosition> , <endPosition> , <timeout> , <flicker>)

Implementation

Click <startPosition> and <endPosition> to define the start position and end position that determine the boundaries of a rectangle inside which it is possible for the string to appear. The start position is mandatory, and the end position is optional. Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: For Screen Id

Waits for a specific screen ID and all of the screen's identification strings to appear.

Format

Wait for screen (<all expected screens>, <timeout> , <flicker>)

Implementation

Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: While String

ApplinX Server waits while the screen displays the string of characters. The server realizes that all the data has arrived when the string is no longer displayed on the screen.

Format

Wait while string (<string> , <startPosition> , <endPosition> , <timeout> , <flicker>)

Implementation

Select the Case sensitive checkbox to determine that the specific string is case sensitive. Click <string> to define the specific string. Click <startPosition> and <endPosition> to define the start position and end position that determine the boundaries of a rectangle inside which it is possible for the string to appear. The start position is mandatory, and the end position is optional. Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: While Cursor

ApplinX Server waits while the screen displays the cursor in a particular position. The server realizes that all the data has arrived when the cursor is no longer displayed on the screen.

Format

Wait while cursor (<startPosition> , <endPosition> , <timeout> , <flicker>)

Implementation

Click <startPosition> and <endPosition> to define the start position and end position that determine the boundaries of a rectangle inside which it is possible for the string to appear. The start position is mandatory, and the end position is optional. Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: While Screen Id

The ApplinX Server waits while the screen is displayed. The server realizes that all the data has arrived when one of the screen's identification strings are no longer displayed.

Format

Wait while screen (<timeout> , <flicker>)

Implementation

Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Wait type: While dynamic string

Wait while dynamic string (<startPosition> , <endPosition> , <timeout> , <flicker>)

Implementation

Click <startPosition> and <endPosition> to define the start position and end position that determine the boundaries of a rectangle inside which it is possible for the dynamic string to appear. The start position is mandatory, and the end position is optional. Click <timeout> to determine the amount of time in milliseconds, that the ApplinX Server should wait for the screen to arrive. 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. Click <flicker> to define this additional amount of time.

Test Screen Name

Checks whether the selected screen matches the current host screen.

Implementation

Select a screen to test.

Test Field Attribute

Boolean test of a selected attribute in the current screen/screen group, according to position or field.

Format

```
Is/Is not <Field Attribute> in Position (row, column)

Is/Is not <Field Attribute> in Field(<select field>)
```

Implementation

1. Click Is/Is not to define the condition.
2. Select an attribute: protected, hidden, intensified, reversed video, background color, foreground color, content, blinking or underlined.
3. Select whether to retrieve the attribute of a specific position (specify the row and column) or application field (select a field).

Test Screen Content

Checks whether a certain text is within a defined rectangle or field in the current screen/screen group.

Format

```
Is/Is not Content(text) in Rectangle(Start Position:(row, column), End Position:
(row, column))

Is/Is not Content(text) in Position(row, column) (length)

Is/Is not Content(text) in Field(<select field>)
```

Implementation

1. Click Is/Is not to define the condition.
2. Click on `<text>` to define the relevant text.
3. Select whether the text is in a specific position (specify the row and column and length of the string), rectangle (specify the start and end row and column) or field (select a field).

Is Member of Group

Determines whether the current screen is a member of a specific Screen Group.

Format

`Is/Is not Screen member of (Screen Group).`

Implementation

1. Select the relevant Screen Group.
2. Click Is/Is not to define the condition.

Host Keys

A free text editor that allows selecting host key constants.

Implementation

Select a host key and then add tokens to use values from the context. Use replaceable tokens, using the ApplinX format `$(<Replaceable Token Name>)`. Click **Finish**. Map the host key expression to the relevant output.

Example

Enter text and replaceable tokens:" Your account number is - $(number)". `$(number)` being a replaceable token.

Position

A row and column structure that defines a specific position.

Format

`Position( <row> , <column> )`

Implementation

Define the position by clicking the row and column expressions.

Screen Name

Returns the name of the current host screen.

Field Attributes

Returns the attributes of a position or field in the current screen/screen group.

Format

```
Get Attributes from Position(row, column)

Get Attributes from Field (<select field>)
```

Implementation

Select whether to retrieve attributes from a specific position (specify the row and column) or field (select a field).

The attributes that are retrieved are:

- Content - the content of the specific position (according to the defined length), or field.
- isProtected - True when protected, and False when unprotected.
- isIntensified - True when intensified, and False when not intensified.
- isHidden - True when hidden, and False when not hidden.
- isReversedVideo - True when reversed video, and False when not reversed video.
- isBlinking - True when blinking, and False when not blinking.
- isUnderlined - True when underlined, and False when not underlined.
- backgroundColor - returns the background color code and name (see list below).
- foregroundColor - returns the foreground color code and name (see list below).

Color	Code #		Color	Code #
None	-1		Gray	8
Black	0		Light blue	9
Blue	1		Light green	10
Green	2		Light aqua	11
Aqua	3		Light red	12
Red	4		Light purple	13
Purple	5		Yellow	14
Brown	6		Light white	15
White	7			

Note: These color codes are consistent with the codes that the Base Object fields return.

Screen Properties

Enables retrieving the following screen properties:

- The row, column and/or field where the cursor is positioned.
- The width and height of the screen.
- Indication whether the screen is a window or not.
- Retrieves the window title.

Screen Buffer

Retrieves the screen content as an array, where each index in the array is a row of the screen. Determine whether to retrieve the window content only or the whole screen.

Format

```
Get Screen Buffer(onlyWindowBuffer)
```

Implementation

Click onlyWindowBuffer to define a boolean expression which will determine whether to retrieve just the window content or the whole screen.

Field Content

Returns text from the current screen/screen group according to a position or field.

Format

```
Get Screen Content from Position(row, column)

Get Screen Content from Field (<select field>)
```

Implementation

Select whether to retrieve text from a specific position (specify the row and column) or field (select a field).

Find Field Index

Finds the index of the first occurrence, within a multiple application field, that matches a specified criteria. Select the relevant multiple field and the type of attribute to search for, and then define the criteria.

Format

```
Find the index where(Is/Is not <field attribute>(<comparison type> <text>) in
Field (<select field>))
```

Implementation

1. Click Is/Is not to define the condition.

2. Select an attribute: protected, hidden, intensified, reversed video, background color, foreground color, content (define whether it is exact text or whether the field contains the text), blinking or underlined. According to the selected attribute, you may be required to determine further search parameters such as colors, the specific screen content etc.)
3. Select the relevant multiple application field.

Field Occurrences

Returns the number of instances of a multiple application field in the current screen\screen group.

Format

```
Count of Field(<select field>)
```

Implementation

Click on select field. Select a specific screen\screen group and then select a field from the list of fields.

Web Procedure Specific Expressions

Web Procedure Expressions

- Test Web Element Exists
- Web Element Content

Test Web Element Exists

Boolean test of a selected XPath in the current page. This function does not wait for all the elements of the page to be loaded and is carried out straight away.

Format

```
Is/Is not Web Element (<XPath>) exists
```

Implementation

1. Click Is/Is not to define the condition.
2. 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).

Web Element Content

Returns the content of the element according to the selected XPath. If the element is not found after a certain amount of time (by default 30 seconds), this expression will timeout. This time can be changed in the <ApplinX installation>/config/gxconfig.xml file:

```
<MainConfiguration>
   ...
    <ServerConfiguration>
      ...
         <webProcedureConfig>
              <waitElementTimeout>30000</waitElementTimeout>
         </webProcedureConfig>
    </ServerConfiguration>
</MainConfiguration>
```

Format

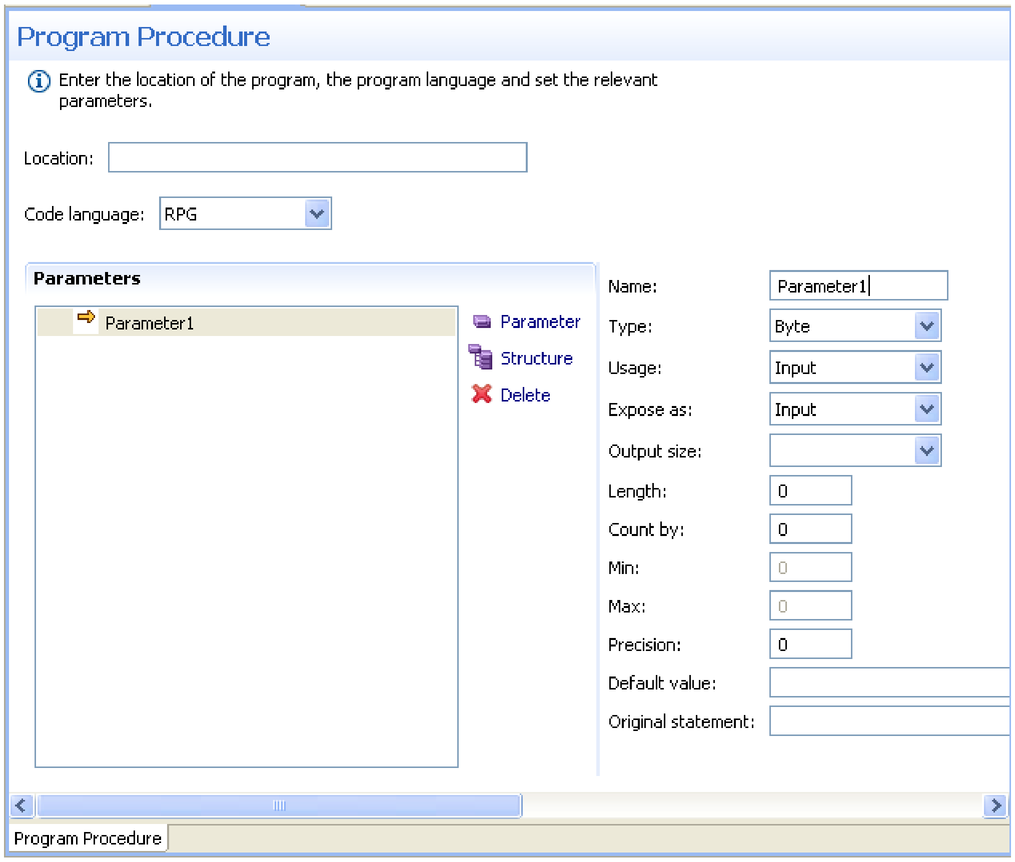
```
Get Content from Web Element (<XPath>)
```

Implementation

Select the XPath from which to retrieve the content.

Program Procedures

Double-click on the relevant Program Procedure from within the Repository. The Editor area displays the Procedure Program.


Program Procedure

Enter the location of the program, the program language and set the relevant parameters.

Location:

Code language: RPG

Parameters

Parameter1

Name: Parameter1
Type: Byte
Usage: Input
Expose as: Input
Output size:
Length: 0
Count by: 0
Min: 0
Max: 0
Precision: 0
Default value:
Original statement:

Location
: Fully qualified path of program in the host file system.

Code language
: RPG or COBOL.

Name
: Name of parameter

Type
: Data type of parameter

Usage
: Select the parameter usage: input, output, or input/output.

Expose as
: Select which parameters are exposed to the end user. By default, a parameter is exposed according to its usage (input, output, etc.)

Length
: Enter the parameter's length.

For RPG use only: You may either type a number or select from the drop-down list a reference to another numeric parameter.

Output size (optional)
For RPG use only: If the output size for this parameter is different from its length (array etc.), enter the expected output size. You may either type a number or select from the drop-down list a reference to another numeric parameter.

Count by (optional)
If this parameter is an array, enter the number of members in this array. Either type a number or select from the drop-down list a reference to another numeric parameter for dynamic arrays. For dynamic arrays you may also specify the minimum or maximum array boundaries.

Default value (optional)
Enter a default value for this parameter.

Precision (optional)
For numeric parameters, enter the number of digits that follow the decimal point.

Original Statement (optional)
If imported by the program wizard, this box will display the original statement from the program's source file for this parameter.

9 Connection Pool

See also Connection Pools under *Designing and Developing an Application*.

General

Log level
It is possible to set a different detail of logging for each connection pool. The server logger should be set to **Info** log level at least, in order to see connection pool logs.

Initialization mode

Manual
An administrator manually initializes the connection pool.

When first accessed
Web applications or sessions that connect to the connection pool on the first call.

Automatic
Automatically initializes the connection pool when the application is loaded.

Pool

Disconnect after usage
When the check box is selected, a connection will not be used again after it is returned to the pool. The termination path will be executed and then the connection will be disconnected. A new connection will be initialized if needed, according to the connection pool size definitions.

When the check box is not selected, the recycle path (it is recommended to define a recycle path when Disconnect after usage is not selected) is performed. The termination path will be executed once the connection pool stops.

User wait for connection timeout
The time, in milliseconds, that a user should wait to get a connection from the server before giving up. Leaving the box empty (or 0) sets no timeout at all.

Delay between connection attempts
The minimum delay time, in milliseconds, between any two consecutive connection creation attempts. Leaving the box empty (or 0) sets no delay at all.

Pool size type
Fixed: Constant number of connections (no pool size policy).

Limited: Number of connections is bound in a range.

Flexible: Number of connections is not bound by the pool.

Number of connections
When selecting Fixed as the size type, select the fixed number. When selecting Limited or Flexible enter the minimal and maximum number of connections in the pool as required.

Available connections
Not relevant when selecting Fixed as the size type. This field determines the pool size policy, see **Pool Size Policy Considerations**.

Minimal number of available connections in the pool. If the number of available connections in the pool falls below this number, new connections will be created until the minimum is reached.

Maximal number of available connections in the pool. If the number of available connections in the pool exceeds this number, some connections will be terminated until the maximum is reached.

Keep-alive

Path
A path that keeps the connection at the connection pool initial screen. Use the folder selection icon to select paths from different folders.

Interval
The time, in minutes, that an available connection may stay idle. When this time elapses, a keep-alive path is executed.

Connection information set

Select set
Select an existing **Connection Information Set**, or create a new one by selecting "<new set>" and pressing the button, labeled "New". To edit existing information set, select it in the combo box, and click the button (now labeled "Properties").

Delay after using a set
This delay takes place after a connection is terminated, and before a new connection can be initialized using the same parameter set. However, the delay will not occur if the connection information row has an unlimited repeat value (0). This feature is required for cases where the host can't initialize a new connection immediately after closing the current one, using the same parameters (user and password, device name, etc.).

Note: Changes to the pool's parameters will take place immediately, though the pool may take a while to fully adjust to the new configuration. Furthermore, changes will not cause the sudden disconnection of current active users.

Pool Size Control Policy Considerations

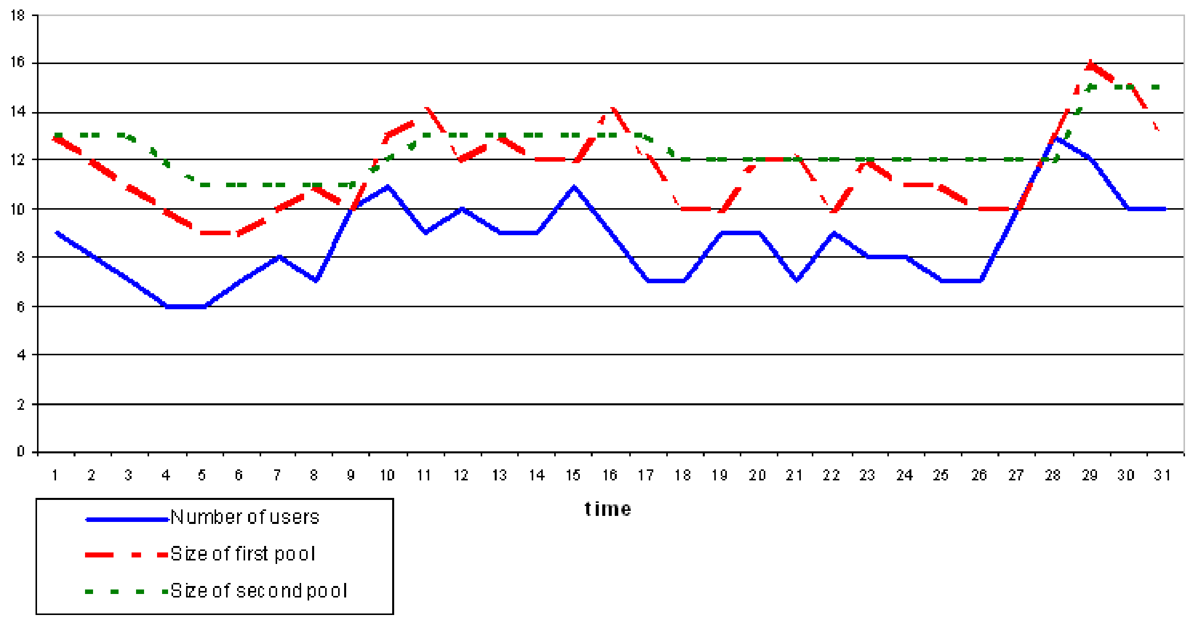
A simple (but not recommended) size control policy would be to set the minimal and maximal number of available connections with the same value. Under such a policy, the pool will try to always have a fixed number of available connections at hand. Therefore, when a client receives a connection, the number of available connections is reduced by one. The pool will initialize a new connection. When the client returns a connection, the pool recycles it and then it has one extra available connection. The pool will then terminate one of its connections to maintain the fixed number of available connections.

As far as the host server is concerned, every user that used the connection pool requires a new connection and initialization process (besides the constant set of "buffered" connections). However (unless working with a disconnect after usage policy), other pool size policies can diminish the amount of work required from the host server for the same amount of users.

Let n be the minimal, and n+k the maximal numbers of available connections. Usually there will be n to n+k available connections at hand. When a client receives a connection, a new connection will only be created if there were exactly n available connections. Similarly, a connection will be terminated after a connection is returned to the pool only when there are already n+k available connections in the pool.

Let d be the maximal absolute difference between the number of connections taken from the pool and the number of connections returned to it during every moment in a given period. As long as d < k no new connections to the host server are required, no matter how many users are served during that time.

The chart provides an example of the possible effect of pool size policy. The blue line represents the number of users connected at any moment. The first pool is configured to have three available connections at any given time. The second pool has a range of available connections: two (minimum) to five (maximum). The size of the first pool (red line) varies exactly according to the activity of the users. It requires the host to initialize and logoff 19 additional connections while serving 40 users. The second pool is more stable in size (green line), and requires only five additional connections and destroys only three, serving the same amount of users.



To conclude, we see that using a range of available connections (as opposed to holding a fixed size buffer) can dramatically reduce the number of connections and disconnections from the host server.

Note: This is true only when using a recycle path instead of disconnect after usage policy.

Connection Information Sets may affect the actual maximum size of the pool. For example when a limited number of users and passwords with a repeat limit of one is defined in the connection information set, the maximum number of possible connections will be the number of users defined in the connection information set even if the defined pool size is flexible. Refer to Connection Information Set for further details.

Navigation

Initialization

A path that can be executed on any new connection to the host that navigates the connection to one of the Initial Screens.

Initial Screen

The initial screen is the first screen after the Initialization Path ends. The connection pool initial screen can be selected from the identified screens list. The Application Map can be used for this purpose.

Recycle

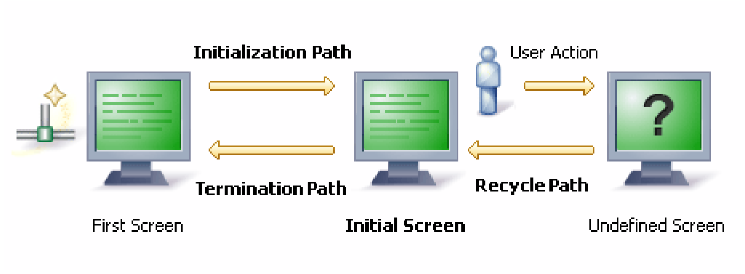
A path that can be executed on any used connection returned to the pool that navigates the connection to one of the Initial Screens.

Run recycle anyway

When checked, recycle path will always be performed after the connection has been used, even if the connection is in the initial screen.

Termination

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).



10 Web Services

External Web Services

Note: This feature is only available when working with a SOA license.

Service Name
The name of the External Web Service.

Port Name
The name of the selected port.

Namespace
The target namespace of the Web service.

Endpoint URL
The URL to which the SOAP client sends requests to activate the service.

WSDL URL
The URL of the WSDL which describes the Web service.

Change Target Server
Changing the target server enables replacing the server address and port. 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.

Once this button is clicked, the *Changing the Target Server* dialog box is displayed:

Address
The new address (IPv4 and IPv6 address formats are supported).

Port
The new port number.

Update WSDL URL
When relevant, you can select this check box and update the WSDL URL.

Apply to other External Web Services that use the same target server
You can select to apply these changes to all External Web Services that use the same target server (the list of updated services appears in the server log). When this option is not selected, only the current External Web Service will be changed.

See also: Using External Web Services

11 Connection Information Sets

Table Area

There are two initial columns and one row in the table. The ID column indicates <default> for the initial row (cannot be deleted). Other rows are automatically assigned a numeric Id. The Repeat column can be edited (but cannot be deleted), and states the maximum number of host **connections** that can use the information row simultaneously. More columns and rows can be added to the table.

Add Record

Adds a new row to the table. Row ID is set automatically.

Delete Record

Deletes selected row or rows. The default row cannot be removed.

Define Columns

Enables defining a column in the table. The Column Selection dialog will open to allow you to select or remove columns.

In the column selection dialog box, entities can be added as table columns: connection parameters, application fields and variables. Connection parameters are parameters that are required for initializing a session with the host such as the device name or host name. The parameters defined here override the general parameters defined in the application configuration. Application fields and variables provide the data required by the initialization path that is used in connection pools. Use application fields or variables in accordance with the requirements and definitions of the relevant path. When the initialization path is a path procedure, you can only use variables to provide the required data. Many times one of the columns contains password information. It is recommended to protect such the values entered in this column by applying the Password column feature.

Set Password

Sets the column as a password column, making it protected.

Set Default Values

It is possible to set default values for the parameters/variables in the Connection Information Set. When clicking this link, a dialog box is displayed enabling you to enter default values for each of the columns.

12 Data Structure

Attribute Types

- Text
- Long
- Boolean (True, False)
- Double
- Integer
- Float
- Byte
- Date: The format can be either 2001-07-04 12:08:56.235 or 2001-07-04T12:08:56.235 (this format should be used for Web Services only), or dd/MM/yyyy

Refer to Data Structure for information regarding this entity.

13 Database Entity

Available in SOA applications only.

Folder
The folder where the database is located.

Username
The username required to connect to the database (optional).

Password

URL additional parameters
Additional parameters needed for the database connection (optional).

Driver
The connection driver used to access the database. You can choose a pre-defined driver using the combo box or supply a custom one. It may be required to supply the driver itself to ApplinX Server.

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

URL
The connection prefix required to connect to the database. Choose a pre-defined prefix from the combo box or supply a custom one.

14 Business Activity

Summary File (CSV) of Trace File Analysis

Every time the business process activities are extracted, a CSV file is created summarizing the extraction process/contents. This file is created in a newly created directory called "AnalysisSummary" in the application root directory. The CSV file is called "AnalysisSummary_<timestamp>.csv".

Parameter name
Trace File Name
Output File location
Activity entity Name
Start time
End time
ProcessId
ActivityProcessor
Name
DisplayName
ProcessStepId
PreviousStepId
User defined $i

Every line includes the following:

III ApplinX Visual Studio .NET Add-In

15 ApplinX Visual Studio .NET Add-In

ApplinX Visual Studio add-in enables generating procedures as detailed in Exposing Procedures. The add-in generates a Procedure Client as a C# class or a .NET Web Service or creates Web References to Web Services.

The add-in enables generating the following:

Installation

Prerequisites:

Visual Studio 2008/2010.

To install ApplinX VS .NET

1 Begin the ApplinX installation process as detailed in the Installation section.

2 In the screen where you select the components to be installed, ensure that you select Visual Studio Add-in.

3 Complete the ApplinX installation.

Working in the VS.NET Development Environment

To Work with ApplinX VS.NET add-in in the VS.NET environment:

1 From the **Tools** menu, select **Add-In Manager**. The Add-In Manager dialog box appears.

2 Check "ApplinX.NET Add-In". The ApplinX Explorer window will be displayed.

Note: The above process is only required the first time you use the add-in, thereafter select **ApplinX Explorer** from the **Tools** menu. ApplinX Explorer will appear on the left.

Adding an ApplinX Server

By default, the add-in is connected to ApplinX server on "localhost", port 2380. It is possible to configure additional ApplinX servers.

To add an ApplinX server:

1 In the ApplinX Server Explorer window, focus on the Legacy Connections node.

2 Click the **Add new ApplinX Server** icon in the toolbar or select **Add new ApplinX Server** from the right-click shortcut menu. The New ApplinX server dialog box will appear.

3 Enter a `server name`, the server `IP address` (IPv4 and IPv6 address formats are supported) and the `HTTP server port`.

4 Click **Save**. The server will appear in the tree, with a list of the ApplinX applications, which are defined on this server.

To delete a server, select the server name in the ApplinX Server Explorer, click the **Delete ApplinX server** icon in the toolbar, or select **Delete ApplinX Server** from the right-click shortcut menu.

Web Enablement: Generating a Screen aspx Page and a Procedure Client

To generate an aspx Web Page based on an ApplinX Screen

1 Expand the Web Enabling node, and select the relevant screen.

2 Right-click on the screen and select Generate ASPX Page.

To generate a Procedure Client

1 Expand the Integration node, and select the relevant Procedure Group.

2 Right-click on the procedure group and select **Generate Procedure Client**.

SOA Enablement: Using a Procedure Client

To generate a Procedure Client:

1 Expand the Integration node, and select the relevant Procedure Group.

2 Right-click on the procedure group and select **Generate Procedure Client**.

3 A class will be created, based on the procedure group's name. A reference to *GXBinaryExecuter.dll* file will be added (The DLL itself will be copied to the Project's bin directory). This DLL is mandatory in order for the Procedure Client to work.

To generate a Procedure Client ASPX Page:

1 Expand the Integration node, and select a Procedure within the relevant Procedure Group.

2 Right-click on the procedure group and select **Generate Procedure Client ASPX Page** (relevant only to Web Site projects).

IV Appendix

16 Appendix A: Reserved Words in ApplinX

There are words which cannot be used as the name of an entity or application:

- abstract
- addressof
- App_Code
- App_Data
- as
- asp
- aspx
- assert
- base
- bool
- boolean
- break
- byte
- case
- catch
- char
- checked
- class
- clear
- close
- config

- const
- context
- continue
- cs
- css
- date
- decimal
- default
- delegate
- delete
- dim
- do
- document
- double
- else
- end
- enum
- escape
- eval
- event
- exception
- explicit
- extends
- extern
- FALSE
- field
- final
- finally
- fixed
- float
- for
- foreach
- friend

- function
- get
- goto
- handles
- if
- images
- implements
- implicit
- import
- in
- index
- inherits
- instanceof
- int
- interface
- internal
- is
- java
- js
- jsp
- lock
- long
- me
- mustoverride
- mybase
- myclass
- namespace
- native
- new
- next
- nothing
- notoverridable
- null

- object
- open
- operator
- out
- overridable
- override
- overrides
- package
- page
- params
- private
- protected
- public
- put
- readonly
- redim
- ref
- return
- sbyte
- screen
- sealed
- shadowing
- short
- sizeof
- stackalloc
- static
- strictfp
- string
- struct
- sub
- substring
- super
- switch

- synchronized
- target
- this
- throw
- throws
- tostring
- transient
- TRUE
- try
- typeof
- uint
- ulong
- unchecked
- undefined
- unsafe
- until
- ushort
- using
- value
- var
- vb
- virtual
- void
- volatile
- while
- window
- withevents
- write

17 Appendix B: Security in ApplinX

End-To-End Security

ApplinX multi-tier architecture supports end-to-end security by utilizing encryption and industry-standard, secured protocols within each layer of communication. The following document details the security measures that are available for each layer, as well as additional security mechanisms available in other ApplinX components.

ApplinX server and clients support the ciphers defined at JVM level. See list of supported SSL cipher suites.

For SOA and Web Enablement Solutions

Host <> ApplinX Server

Communication between ApplinX Server and the host can be encrypted using SSL V3. Both client and server authentication are supported. SSL X509 certificate is stored using standard Keystore implementations (JCEKS).

This feature is available for any host that supports SSL V3 communication, however, this has only been tested on Mainframe hosts. It is also possible to use the secured protocol SSH V2 (instead of the VT protocol).

To configure an SSL connection between the host and ApplinX server:

- Refer to Configuring the SSL Connection.

For Web Enablement Solutions

ApplinX Server <> ApplinX Base Object/ApplinX Web Applications

Communication between ApplinX Server and the ApplinX Base Object (which resides on the web server or application server) can be encrypted using SSL (this layer of security includes encryption only, without authentication).

To configure an SSL connection between ApplinX server and the Web Server/Application Server:

1. In the Server Properties, General tab, select the **Secured port** checkbox.
2. In the Web Application Configuration Manager set the Session server URL to the SSL port (e.g. applinxs://localhost:23443).

It is possible to set the address dynamically from within the Base Object by using the method setServerURL in the ApplinX SessionConfig object within the GXBasicContext file, gx_initSessionConfig method (IPv4 and IPv6 address formats are supported).

Note: When you want to enable connecting to the HTTPS port only from the ApplinX server machine, add the following system variable when starting the server:
`-Dcom.sabratec.applinx.securesocket.localonly=true`

Web Server / Application Server <> End User Browser

Communication between the end user web browser and the Web server or application server can be secured using HTTPS, or a firewall. This feature is not related to ApplinX and should be supported by the Web Server / Application Server.

The ApplinX .NET Framework also supports authentication of end users using Integrated Windows Authentication (formerly referred to as NT Authentication), as this is a built-in feature in the Microsoft .NET environment.

For SOA Solutions

ApplinX Server <> Procedure Clients

The Java and .NET Procedure Clients (binary SOA clients), can be encrypted using HTTPS (SSL).

To configure an SSL connection between ApplinX server and a Procedure Client

- Follow the instructions in the following topic: Creating a Secure SSL Connection between a Procedure Client and ApplinX Server.

  Note: When you want to enable connecting to the HTTPS port only from the ApplinX server machine, add the following system variable when starting the server:
  `-Dcom.sabratec.applinx.securesocket.localonly=true`

ApplinX Server <>Web Services (WS-Stack)

To configure an SSL connection between ApplinX server and a Web Service (WS-Stack)

- In the Server Properties, General tab, select the **Secured port** checkbox. ApplinX will automatically connect to WS-Stack using a secure connection.

  Note: When you want to enable connecting to the HTTPS port only from the ApplinX server machine, add the following system variable when starting the server: `-Dcom.sabratec.applinx.securesocket.localonly=true`

Web Services (WS-Stack) <> Web Service Client

This layer can be encrypted using HTTPS (SSL). Refer to **http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html**and also to the relevant commented section in <ApplinXInstallationDirectory>/conf/server.xml.

Development Time

ApplinX allows managing password-protected users, groups and their permissions. It is possible to define certain permissions to a group, and then associate users with this group, giving the user the permissions defined for this group or to define specific users permissions. Each user/group can be assigned with read/write permissions at the application or folder level. The users' definitions are saved in an encrypted configuration file.

It is also possible to define users based on Integrated Windows Authentication (formerly NT Authentication).

Connection Pools

It is possible to specify passwords of host users as part of the connection information sets of connection pools (to enable connection pooling with automatic login to the host application). These passwords are encrypted and saved in the application's repository database.

Running ApplinX Server with a Java Policy File

In order to run the ApplinX server with a Java security manager enabled, the following flags should be appended to the Start_Process_Parameters in the <ApplinX installation>\bin\start-gx-server.bat file, or to the JAVA_OPTS in the <ApplinX installation>\bin\start-gxserver.sh file or to the Start_Process_Parameters in the GXApplinXService.ini file:

```
-Djava.security.manager -Djava.security.policy=./conf/catalina.policy
```

In the policy file (specified in the path above) the following permissions are set inside a grant section (if a different policy file is used, one should add the following manually):

```
permission java.net.SocketPermission "localhost:2323" , "listen,resolve,accept";
permission java.net.SocketPermission "localhost:*" , "resolve,accept";
permission java.net.SocketPermission "<host name>:<host port>" , "connect,resolve";
permission java.io.FilePermission "${com.sabratec.gxhome}/-", "read, write, delete";
permission java.io.FilePermission "${catalina.home}/-", "read";
permission java.io.FilePermission "${java.home}/../-", "read";
permission java.io.FilePermission "${java.io.tmpdir}/" , "read, delete, write";
permission java.io.FilePermission "${java.io.tmpdir}/-" , "read, delete, write";

//ApplinX Xstream usage. Used mostly by ApplinX configuration persist to XML
permission java.lang.RuntimePermission "accessClassInPackage.sun.misc";
permission java.lang.RuntimePermission "accessClassInPackage.sun.reflect";
permission java.lang.RuntimePermission "accessClassInPackage.sun.io";
permission java.lang.RuntimePermission "accessClassInPackage.sun.logging.*";
permission java.lang.RuntimePermission ↩
"defineClassInPackage.org.apache.jasper.runtime";
permission java.lang.RuntimePermission "accessDeclaredMembers";
permission java.lang.RuntimePermission "createClassLoader";
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
permission java.lang.RuntimePermission "reflectionFactoryAccess";
permission java.io.SerializablePermission "enableSubclassImplementation";
permission java.lang.RuntimePermission "getClassLoader";
// For using Log4J
permission java.lang.RuntimePermission "defineClassInPackage.java.lang";

// Used for showing the server icon in the system tray. Uncomment if needed.
// permission java.lang.RuntimePermission "loadLibrary.GXUtil";
// permission java.lang.RuntimePermission "modifyThreadGroup";

permission java.io.SerializablePermission "enableSubstitution";
permission java.sql.SQLPermission "setLog";
permission java.util.PropertyPermission "com.sabratec.*", "read,write";
permission java.util.PropertyPermission "com.softwareag.*", "read,write";
permission java.util.PropertyPermission "*", "read";
permission java.util.PropertyPermission "org.apache.adb.properties", "read,write";
permission java.util.PropertyPermission "javax.xml.registry.ConnectionFactoryClass", ↩
"write";
```

To allow ApplinX to integrate with WSStack, the following are needed:

```
permission java.net.SocketPermission "<ip of host to where the WSStack is ↩
deployed>:*", "accept,resolve";
permission java.net.SocketPermission "<machine name to where WSStack is deployed: ↩
WSStack Http port>", "connect,resolve";
permission java.lang.RuntimePermission "getProtectionDomain";
permission java.lang.RuntimePermission "modifyThread";
permission java.lang.RuntimePermission "getenv.WS_STACK_HOME";

// In case WSStack is run in the server's process, the following are also needed
permission java.net.NetPermission "specifyStreamHandler";
permission java.lang.RuntimePermission "shutdownHooks";
permission java.lang.RuntimePermission ↩
"defineClassInPackage.org.apache.jasper.runtime";
permission java.lang.RuntimePermission "modifyThread";
permission java.lang.RuntimePermission "setContextClassLoader";
```

Note: lines with a close that starts with a single '<' character, should be edited according to the text inside the close.

When ApplinX is running with SSL support, the following should be added as well:

```
permission java.io.FilePermission "${java.home}/jre/bin/keytool" ,"execute";
```

Blocking Access to URLs in an ApplinX Web Application

You can block access to each URL of a web application by using a property file that contains the URLs of the web application pages and the access to each URL. The following access roles are available:

- **Unauthenticated**
  User that did not pass the authentication process and has the minimal access permission.
- **User**
  User that has regular permissions
- **Admin**
  User with the highest permissions.

To enable blocking of selected URLs

1 Uncomment the `filter` section in file *web.xml* of the web application.

```
<!-- <filter>
    <filter-name>GXCheckURLProxyFilter</filter-name>
    ↩
<filter-class>com.sabratec.applinx.j2ee.framework.web.filters.GXCheckURLProxyFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>GXCheckURLProxyFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping> -->
```

2 Update the property file *URLPermissions.properties* to set the URL permission. Asterisk notation is supported for directories and file extensions.

3 Set the role of the user in a session by setting the method `setSessionRole` in class inherited from `GXAbstractWebContext`.

The table below shows the default access for the URLs of ApplinX applications:

Resource Pattern	Role
`z_resourceReader.jsp`	All (Unauthenticated)
`run_printlet.jsp`	All (Unauthenticated)
`config/*`	Administrator
`log/*`	Administrator
`z_editConfig.jsp`	Administrator
`.class`	Administrator
`Index.jsp`	All (Unauthenticated)
`__PREVIOUS_VERSION`	Administrator

All URLs not defined in the property files can be accessed only by an authenticated user. When one file is set to two roles, the last definition in the property file has priority.

By using asterisk notation to define directories and/or group of files for a role (e.g. `config/*`) you can exclude one file from the group by providing a more exact definition (e.g. `config/web.xml`).

This feature is available when creating a new web application. If you want to use this feature for an existing web application, you will need to manually add the filter code to the file *web.xml* and also add the property file *URLPermissions.properties* to the application.

18 Appendix C: Natural for UNIX Installation

Installing APXcomponent for Natural UNIX Protocol

When the host system on which you run your Natural applications is a UNIX system, additional software for the Natural UNIX protocol has to be installed on the host. The Natural UNIX protocol modules are shipped together with Natural for UNIX.

The protocol enables:

- The option to change the host password via the ApplinX Base Object.
- Support for a non standard screen size.
- User permissions are now received directly from the host. Note that currently it is not possible to log in to the host without providing a user name and password unless you use a customized User Exit authentication function.
- Host keys are automatically recognized by ApplinX and do not need to be identified according to their pattern.
- It is now possible to define the authentication method and authentication localization texts.
- The configuration has been improved. It is possible to specify the Natural parameter file in ApplinX Designer, using the same port for a number of applications in parallel, without making changes in the host.

Prerequisites

Supported Operating Systems: The same platforms as those supported by Natural for UNIX.

Other Software Products: Natural Version 6.3.2 or above for UNIX.

Linker: A linker (for example, lD or cc) and the command make must be available in the system.

Setting up the APX Component

Setting up APX on UNIX consists of the following steps:

1 **Stop the NAT UNIX apxsrvd Daemons**

> **Note:** This step is only required for an upgrade installation. It is not required when you install Natural for UNIX for the first time.

Stop the apxsrvd process using one of the following commands: `apxsrvd.sh servicename stop`

or

```
apxsrvd.sh portnumber stop
```

Repeat this command for each service that has been started.

2 **Establish the Environment**

Ensure that the environment definitions, as described in setup.txt in the root directory of the Natural CD, are correct and set.

> **Note:** Special note for Red Hat Enterprise Linux AS: On Red Hat Enterprise Linux AS, only PAM authentication, working with local-machine users is supported (for example it is not possible to use PAM with WinBind for active directory authentication. Implement **UserExit1** to use PAM this way). Therefore you have to create a file */etc/pam.d/apxsrv* containing the following two lines:

```
#%PAM-1.0 auth required /lib/security/pam_unix.so nullok
```

3 **Install Natural with APX Component**

ApplinX Natural UNIX protocol is installed during the Natural installation.

> **Important:** The Natural installation provides a APX option which must be activated. For more details, see your Natural installation documentation.

When you install Natural with APX, the directory $NATDIR/$APXNODE is created. The template files located in $NATDIR/$NATVERS/apx/node-name are then copied to this new directory. The APX daemon $NATDIR/$NATVERS/apx/bin/apxsrvd requires a TCL shared library which is delivered in the $NATDIR/$NATVERS/lib directory. It is linked to the runpath /opt/softwareag/nat/$NATVERS/lib (for Natural Version 6.3) or /opt/softwareag/Natural/v`<version>`/lib (as of Natural Version 8.3), and is installed with permissions 6755 (s-bit). As the s-bit is used, $LD_LIBRARY_PATH is not searched. Therefore, ensure that the apxsrvd daemon can locate the TCL shared library. This can be done by installing Natural in /opt/softwareag and setting a symbolic link from /opt/softwareag to your current $SAG directory, or making the TCL shared library available from a system directory.

4 **Check the Environment Variables for APX**

The APX-specific settings are shown below:

Environment Variable	Description
APPLINX_ROOT	Home directory.
APXNODE	Name of the node on which APX is installed.
APXSERV	Name of the path to the apxservice file.
APXTIMEOUT	Number of seconds that the daemon waits for an input from the ApplinX server side (SO_TIMEOUT).

Directories

The following directories are created when Natural is installed together with ApplinX on a UNIX system:

Directory	Description
$NATDIR	Top-level Natural directory.
$NATDIR/$NATVERS	Directory with all components for the current Natural version.
$NATDIR/$NATVERS/apx	Directory with the APX components.
$NATDIR/$NATVERS/INSTALL	Shell scripts and environment files to required to install the product.
$NATDIR/$NATVERS/apx/bin	APX executable files.
$NATDIR/$NATVERS/apx/node-name	Contains the template files (services.dat, apxservice, etc.).
$NATDIR/$NATVERS/apx/samples/userexit	Contains the files for building the sample user exit.
$NATDIR/$NATVERS/bin/build	Contains the library (libapx.a) to link with Natural.
$NATDIR/$NATVERS/bin/build.tr	Contains the trace library (libapx.a) to link a trace version with Natural.
$NATDIR/$APXNODE	Contains the configuration files (services.dat, apxservice, etc.).

Note: The above table lists only the most important directories and files.

Configuration Files

When the APX installation finishes, the directory $NATDIR/$APXNODE will contain the following configuration files:

Configuration File	Description
apx.sh	Shell script to start the Natural application.
apxsrvd.sh	Shell script to start and to stop the daemon.
apxsrvd.conf	Contains the parameters required to configure the authentication method and the localization text.
Apxenv	Default environment script file for bash.
apxenv.csh	Default environment script file for cshell.

Working with the APX UNIX Component

Any Natural application can be used with ApplinX.

Starting a New Natural Application

To start a new Natural application, proceed as follows:

1. Create a new parameter file using the Natural Configuration Utility (see the Natural documentation) and modify the STACK command as follows: `logon library; startprogram; fin`
2. Locate a service/port number which is not in use.
3. When necessary configure the apx.sh shell script. This script is called from the APX daemon in order to start a Natural session. It has the following content:

```
#!/bin/sh

# Extract the arguments
IP_ADDR=$1
CLIENT_ID=$2
PARAMETERS=$3
CUSTOM=$4

# Trace
#PT_TRACELEV=i6000
#SAGTMP=$HOME/tmp
#export PT_TRACELEV
#export SAGTMP

# Natural Data Transfer
NSWUCI_FD=$UNIUCI_FD
export NSWUCI_FD
```

```
#

$NATDIR/$NATVERS/bin/natapx parm=$PARAMETERS etid=$$ >/dev/null 2>&1
```

apx.sh Script Arguments

The shell script will receive the following arguments:

IP_ADDR
The client IP address from where the session is opened.

Note: If there is a proxy, this will not be the IP address of the ApplinX server workstation. Instead, it will be the IP address of the proxy.

CLIENT_ID
When connecting with the ApplinX server, the value of this argument will always be ApplinX.

PARAMETERS
The Natural parameters file as it appears in Natural, without the file extension. This file is passed to the parm argument in the Natural command line. The value of the PARAMETERS argument is taken from the ApplinX server configuration (as defined in the ApplinX Designer, in the Host Configuration dialog box).

CUSTOM
For future use.

4 When necessary configure the apxsrvd.conf file:

The configuration file apxsrvd.conf contains information that the user exits need for the APX daemon. It includes the following content:

```
[UserExits]
; UserExit1=/FS/sag/nat/apxexuex/userexit1/libapxuserexit1.so

[PasswdArguments]
Parameters=

[PasswdMessages]
EnterOldPassword=Enter existing login password:
NewPassword=New Password:
ReEnterNewPassword=Re-enter new Password:
PasswordSuccessful=passwd: password successfully changed for*
```

Sections in Configuration file

[UserExits]
The following user exit can be defined:

UserExit1

The library that is defined by UserExit1 contains the following function:

```
int apx_CheckUsernameAndPassword(const char *pUsername, const char *pPassword,
const char *pNewPassword, char *pErrorMessage)
```

If the key UserExit1 is defined in the configuration file, the function `apx_CheckUsernameAndPassword` is responsible for checking the user name and password. If a new password is received, user exit 1 is also responsible for changing the password.

When there is an error, the return code of the function must be "0"; in this case, the `pErrorMessage` is returned to the client. When the user name and password are correct, the return code must be a value other than "0".

[PasswdArguments]

The key Parameter is used to define any additional parameter(s) that have to be passed to the `passwd` command. For example:

```
passwd -r ldap
```

[PasswdMessages]

The keys in this section define the messages that are to be returned by the system (passwd command) when a user changes the password. If any of these messages are not identified by the daemon, an error will be returned to the client.

Password Mechanism

The password and new password are encrypted on the client side and decrypted on the UNIX side. A maximum of 8 characters is allowed.

When user exit 1 is active, the user name, password and new password parameters are passed to the user exit. When user exit 1 is not active, the daemon checks whether the user name and password are correct for the system. If a new password is sent, the daemon changes the password by calling the UNIX command passwd.

Starting and Stopping the ApplinX Daemon

The APX daemon is responsible for accepting new sessions. This daemon can be started and stopped using one of the following commands:

```
apxsrvd.sh servicename [start|stop]
```

```
apxsrvd.sh portnumber [start|stop]
```

Note: The daemon must be started on a service/port which is not yet in use.

Troubleshooting

To troubleshoot problems in the APX component's functionality it is possible to define tracing its activity. The daemon and the Natural application with the APX lib can be traced separately. In order to determine which of the two components' activity should be traced follow these guidelines: when the problem seems to be to do with connecting the ApplinX server to the APX daemon (including authentication problems), trace the daemon component. For any other problems trace the Natural application. Note that it is not recommended to use the trace file in the production environment on a regular basis.

Tracing the APX daemon:

1. Go to the $NATDIR/$APXNODE directory.
2. Edit the apxsrvd.sh file.

   Find EXECUTABLE=apxsrvd, and change it to EXECUTABLE=apxsrvd.tr
3. Define the following environment variables (the following is an example of how these should be defined in sh/bash):

   ```
   > PT_TRACELEV=i6000
   > SAGTMP=$HOME/tmp
   > export PT_TRACELEV
   > export SAGTMP
   ```

   The trace files are created in the $SAGTMP directory. In the example above this is $HOME/tmp.
4. Restart the daemon:

   ```
   > apxsrvd.sh servicename stop
   > apxsrvd.sh servicename start
   ```

   or

   ```
   > apxsrvd.sh portnumber stop
   > apxsrvd.sh portnumber start
   ```

Tracing the Natural application with the APX lib:

1. Go to the $NATDIR/$APXNODE directory.
2. Edit the apx.sh file.

   ```
   $NATDIR/$NATVERS/bin/natapx parm=$PARAMETERS etid=$$ >/dev/null 2>&1

   Change natapx to natapx.tr
   ```

   Uncomment (by removing the # from the beginning of the rows) the TRACE section.

The trace files are created in the $SAGTMP directory.

By default it is $HOME/tmp. It will take effect when next connecting to the daemon.

Installing Natural for ApplinX on OpenVMS Hosts

If the host system on which you run your Natural applications is an OpenVMS system, additional software for ApplinX has to be installed on the host. The ApplinX OpenVMS modules are shipped on the Natural OpenVMS CD. In general, ApplinX uses the default system parameter values provided with the OpenVMS system.

Prerequisites

Supported Operating Systems: The same platforms as those supported by Natural for OpenVMS.

Other Software Products: Natural for OpenVMS 6.3.8 or above.

Setting Up the ApplinX Components

Setting up ApplinX on OpenVMS consists of the following steps:

1. Stop the ApplinX Daemons: This step is only required for an upgrade installation. It is not required when you install ApplinX for the first time.

   Stop the apxsrvd process using the following command:

   ```
   stop servicename
   ```

   Repeat this command for each ApplinX service that has been started.
2. Establish the Environment: Ensure that the environment definitions, as described in readme.txt in the root directory of the Natural CD, are correct and set.
3. Install Natural and ApplinX: ApplinX is automatically installed during the Natural installation. When you install Natural and ApplinX, the directory NATDIR:[apxnode] is created, where apxnode contains the system name. The template files located in NATDIR:[natvers.APX] are then copied to this new directory.
4. Check the Environment for ApplinX: Besides the logical names NATDIR and natvers as defined by Natural, ApplinX needs the following logical names which are created during the installation of Natural:

Logical Name	Description
apxnode	Contains the system name.
VAXC$PATH	Contains the physical device specification of NATDIR:[natvers.BIN]

Example:

Define VAXC$PATH="ALF9$DKB500:[NATURAL.V41212.BIN]"

In addition, the logical names NATOW and NATFE are redefined during the start of the daemon process to point to the ApplinX images NATAPXnatvers.EXE and NATFEAPXnatvers.EXE.

Optional. If function keys and message lines are to be displayed in their native format (i.e. as normal text), set the environment variable APX_PF_MSG_LINES_NATIVE_FORMAT to "YES":

Define APX_PF_MSG_LINES_NATIVE_FORMAT="YES"

If APX_PF_MSG_LINES_NATIVE_FORMAT is not set or if its value is not "YES", function keys and message lines are detected automatically (default). If they are to be treated in a special way, you have to define the basic rules Function Keys and Message Line in the same way as for a mainframe screen.

5. Check the OpenVMS UAF Parameters: Check the OpenVMS UAF (user authorization file) parameters listed in the table below and, if necessary, modify them. The size values of these parameters for the APXSRVD daemon are important. The sizes depend on the number of sessions that are started by the daemon. Natural error messages will occur if the size limits are reached or if no more quotas are available. In these cases, you have to increase the values so that they meet your needs.

Parameter	Recommended Value
Fillm	Approximately 50 for each session with ApplinX.
Prclm	50
Bytlm	Approximately 30000 for each session with ApplinX.
BIOlm	10000

6. Define the TCP Port Number: The UCX service with the TCP port number must be defined in the system as follows:

   $ UCX SET SERVICE APXDEMO /PORT=22370 /FILE="" /USER="" /PROC=""

   Instead of APXDEMO and the above port number, you can also specify other values. For example, you can create or define the TCP service name APXAPPL1 with the port number 25000.

Directories

The following directories are created when Natural is installed together with ApplinX on an OpenVMS system:

Directory	Description
NATDIR	Top-level Natural directory.
NATDIR:[natvers]	Directory with all components for the current Natural version.
NATDIR:[natvers.INSTALL]	Shell scripts and environment files to install the Natural product.
NATDIR:[natvers.BIN]	ApplinX executable files NATFEAPXnatvers.EXE, NATAPXnatvers.EXE and APXSRVDnatvers.EXE.
NATDIR:[natvers.FNAT]	Contains the old Natural demo application SYSEXAPX for ApplinX.
NATDIR:[apxnode]	Contains the configuration files APXSRVD_servicename.COM, APXSRVD_servicename.LOG, SERVICES.DAT and START_APXSRVD.COM.

The files APXSRVD_servicename.COM and APXSRVD_servicename.LOG are created when the ApplinX daemon APXSRVDnatvers.EXE is started with the procedure START_APXSRVD.COM.

servicename is the UCX service as defined in the file SERVICES.DAT.

natvers indicates the version number and patch level of the corresponding Natural version.

Configuration File

The configuration file SERVICES.DAT is located in the directory NATDIR:[apxnode], where the apxnode contains the node name (for example, NATDIR:[ALF9]SERVICES.DAT).

The content of this configuration file is one line for each defined TCP service:

servicename username natural parm1 ... parmn

servicename	Must be the same name as used in the TCP port number definition (see above).
username	Not used.
natural	This is the program name which must not be changed.
parm1 … parmn	Dynamic Natural parameters.

Example:

```
apxdemo sag natural parm=mypar bp=bp1
apxapp1 sag natural parm=app1 bp=bp1
apxapp2 sag natural parm=app2 bp=bp2
```

Note: If the APXSRVDnatvers daemon does not detect Natural's dynamic parameter ETID, the daemon automatically adds the ETID to the list of the dynamic parameters to be passed to Natural. The ETID added by the daemon has the format ETID=number_ username. It is truncated if the string exceeds 8 characters.

Setting Up and Activating the APXSRVD Daemon

The BYPASS privilege must be authorized for the account to start the ApplinX daemon. The BYPASS privilege must also be set for the daemon process.

When TCP port number and service have been defined (UCX) and the SERVICES.DAT template file has been modified according to your requirements, you can start the APXSRVD daemon to use ApplinX.

To start the daemon, invoke the DCL procedure START_APXSRVD.COM as follows:

@START_APXSRVD.COM service natvers

service contains the name of the service as defined with UCX.

natvers defines the Natural version and patch level.

If both parameters service and natvers are omitted, the defaults APXDEMO and the current Natural version are used. The command procedure creates the temporary file APXSRVD_ servicename.COM which sets up the environment and creates all logicals for ApplinX and starts the daemon.

Once the daemon has been started, the file APXSRVD_ servicename.LOG is created. This file contains information (including the errors) about the daemon.

APXSRVD_ servicename.COM and APXSRVD_servicename.LOG are located in the directory NATDIR:[apxnode].

Note: The account which starts the daemon must hold the privilege IMPERSONATE as the default privilege. It is not sufficient to have an authorized privilege.

Closing the Natural Application and Natural in Error Situations

There are error situations in ApplinX which force the termination of the Natural session. For example, when the user chooses the close button in the upper right corner of the ApplinX window, the communication with the Natural application is disconnected immediately. Then also the Natural process finishes execution immediately without running through the normal close down code of the application. This may lead to inconsistencies in the system, for example, if the corresponding entry for the Natural process remains in the database user queue.

ApplinX returns error codes to Natural which can be used in a Natural error handling routine. To make sure that the close down code of your application is always executed, write an error handling routine which identifies the ApplinX fatal errors or add this code to your existing error handling routine. For critical errors, move the command FIN to the top of the Natural stack in order to finish and close everything correctly.

The ApplinX critical errors are:

6296	Fatal error in the communication. The communication will be disconnected immediately.
6297	Fatal error allocating memory.

Example: Application Program

This Natural program reads the employees file stored in Adabas. The program moves ERRGEN to *ERROR-TA to check any Natural error.

```
DEFINE DATA LOCAL
1 VIS VIEW OF EMP
   2 AA-1                                                                              ↩

END-DEFINE
MOVE "ERRGEN" TO *ERROR-TA
READ VIS
  DISPLAY AA-1
END-READ
END
```

Example: Error Handling Routine ERRGEN

This Natural program with the name ERRGEN checks the Natural error number. If the error number is 6296 (fatal error in the communication), the program stacks the command FIN on top of the Natural stack. The Natural execution finishes immediately and all databases are closed.

```
DEFINE DATA LOCAL
1 ERRNUMBER (N4)
1 LINENUMBER (N4)
1 STATUS (A1)
1 LEVEL (A2)
1 GNPGACTU (A8)
END-DEFINE
INPUT (SG=OFF) ERRNUMBER LINENUMBER STATUS GNPGACTU LEVEL
IF ERRNUMBER=6296
  STACK TOP COMMAND "FIN"
END-IF
END
```

Restrictions

There are several restrictions when using the presentation clients with Natural applications on UNIX hosts:

- **Runtime errors in Natural applications**

  Runtime errors in Natural UNIX applications are handled by the default NSWUCIET error transaction. The user can define another error transaction by setting the *ERROR-TA Natural system variable. See the Natural documentation for details. Sample Natural error transaction:

  ```
  DEFINE DATA
  LOCAL
  1 ERR_INFO
    2 ERR_NR(N5)
    2 ERR_LINE(N4)
    2 ERR_STAT(A1)
    2 ERR_PNAM(A8)
    2 ERR_LEVEL(N2)
  END-DEFINE
  INPUT ERR_INFO
  DISPLAY ERR_INFO
  STACK TOP COMMAND 'E'
  END
  ```

  Starting with Natural 6.1, a default error transaction named NSWUCIET will be used if Natural is running with ApplinX and an error transaction has not been set by the user.

- **Return to the Natural main screen**

  You cannot use Natural applications that return to the Natural main screen. This always leads to wrong screen display and a loss of the session.

- **Natural editors and utilities**

  You cannot use the Natural utilities such as SYSMAIN, SYSDDM and editors such as NATEDIT. This always leads to wrong screen display and a loss of the session.

- **Natural system commands**

  You cannot use any Natural system command such as CATALL, FIND, GLOBALS, HELP, KEY, LIST, SCAN or XREF. This always leads to wrong screen display and a loss of the session.

- **Natural commands SETUP and RETURN**

  You should not use the Natural commands SETUP and RETURN as this may lead to a loss of the session. • Terminal commands Terminal (%) commands are not supported. They do not work when entered in a presentation client.

- **Internal REINPUT**

  The error messages of an internal REINPUT are not displayed in the presentation clients (for example, if you enter the number 500 into a I1 field: NAT1142). Examples of such messages:

NAT1142	Input results in integer value overflow.
NAT1125	Too many significant digits in numeric input value.
NAT1143	Input does not correspond to input edit mask.
NAT1011	Requested function key not allocated.

- **Natural system variable *INIT-ID**

  When using the presentation clients with Natural applications on UNIX hosts, the Natural system variable *INIT-ID will not be filled with a value for the terminal type. Instead, it will contain the value notty.

19 Appendix D: Installing and Setting up Software AG Products under UNIX

This section contains general information which applies when installing and setting up any Software AG product on a UNIX platform.

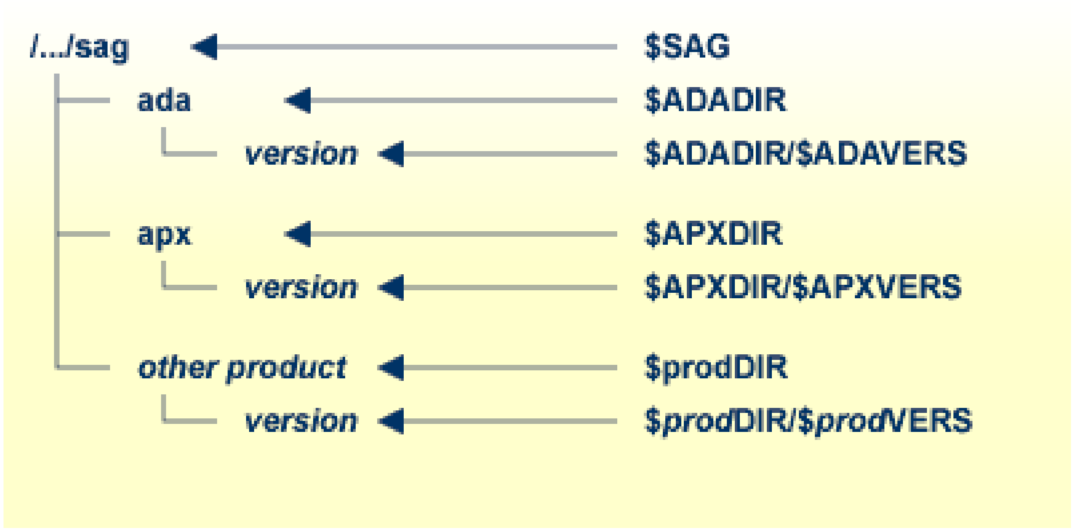
General Information

Installation Package

The installation package containing Software AG products is provided on a CD-ROM conforming to the ISO 9660 standard. The CD-ROM contains a complete directory structure which clearly specifies product and platform.

Software AG Environment

The following figure shows the general directory structure generated during installation and the environment variables which reference the specified directories:



The environment variable $SAG defines the root directory for all Software AG products.

For each product, the variable `$prodDIR` is set to the path of the main directory of the product specified, where prod is a three-letter product code in upper-case letters. For example, all files for ApplinX, whose product code is APX, are contained in the directory `$APPLINX_ROOT`, and for Natural and Adabas, whose product codes are NAT and ADA, the correct directory names are `$NATDIR` and `$ADADIR`, respectively.

The name of the main directory is usually the same as the product code in lower-case letters. For example, the main directory for ApplinX is named apx.

Version-independent parts of the product, such as examples or data, are stored in a subdirectory of the product main directory.

Version-dependent components of the product are kept in the version directory `$prodDIR/$prodVERS`. For example, the current version of ApplinX is stored in the directory `$APPLINX_ROOT/$APPLINX_VER`.

The environment variables `prodDIR` and `prodVERS` for all products specified during installation are defined in the file *sagenv.new*. The same applies for any other environment variables needed for the various products.

Before Installing your Software AG Product

It is recommended to use one common root directory for all of your Software AG product installations. The home directory and the root directory should be separate.

The following activities must be performed if you are installing a Software AG product for the first time, or if your environment has not yet been set correctly due to any other causes.

This section covers the following topics:

- Creating the SAG Administrator's Account and Group
- Backing Up Data
- Logging in as the sag User

Creating the SAG Administrator's Account and Group

You must create one Administrator account and one group for all Software AG products when you install your first Software AG product.

To do this, perform the following steps

1 Define an Administrator account to which all of the Software AG products installed at your site belong. Since all environment definition files for the products are written in Bourne shell syntax, the Bourne (or Korn) shell is recommended as the login shell for the Administrator account. This section assumes that the administrator account is called "sag".

2 Define a group to which the administrator belongs. This section assumes that this group is also called "sag".

3 Create a login directory for the user "sag".

4 Add the group "sag" in the system file */etc/group* and the user "sag" in the system file */etc/passwd*.

Note: To perform these steps, use an appropriate system administration tool.

Backing Up Data

When upgrading a product, it is strongly recommended that you create a backup of your current ApplinX installation and its data using your current product version.

Logging in as the sag User

This description assumes that the user "sag" is the administrator for Software AG products. Log in as the user "sag" (it is not recommended to log in as root).

Installing the Contents of the CD-ROM to Disk

Before performing the following steps, make sure that the administrator user and group have been created and defined.

To install the contents of the CD-ROM to disk

1 Load the CD-ROM in the CD-ROM drive and mount it if this is not done automatically.

Command	Description
`su - root`	To mount a CD-ROM you must be root.
`mkdir /mount-dir`	Create a mount-directory for the CD-ROM.
`mount platform-specific_mount_options device-name /mount-dir`	Execute the mount command (see the table below for operating system-specific mount commands).
`exit`	Return to "sag" user.

2 Platform-specific mount command and options to mount the CD-ROM as ISO9660 or High-Sierra file system:

Platform	Mount Command
AIX	`/etc/mount -v cdrfs -r device-name /mount-dir`
HP-UX	`/usr/sbin/mount -F cdfs -o cdcase device-name /mount-dir`
Solaris	`/usr/sbin/mount -F hsfs -o ro device-name /mount-dir`
Linux (IA-32)	`/bin/mount -r -t iso9660 device-name /mount-dir`

3 Notes: On Solaris, the volume management daemon vold might be active. This daemon mounts the CD-ROM automatically.

Example for Linux:

```
/bin/mount -r -t iso9660 /dev/cdrom /mnt
```

4 Check the directory structure of the UNIX part of the CD-ROM running an ls(1) command on the CD-ROM.

Note: Depending on the mount options used, the files will be all upper case or all lower case. If you mount the CD-ROM as a pure ISO 9660 Interchange Level I CD, you will also see a version number ;1 appended to all files. Please note this for the following steps and use the correct name format.

5 Continue reading the step-by-step installation instructions for the Software AG product being installed.

20 Appendix E: SSL Cipher Suites Supported by ApplinX

Following is a list of the SSL cipher suites supported when connecting to the host. These are relevant when running Oracle's JVM. It is possible to use any JSSE provider, including IBM.

```
TLS_RSA_WITH_AES_128_CBC_SHA

TLS_RSA_WITH_AES_256_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDH_RSA_WITH_AES_128_CBC_SHA

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

TLS_DHE_RSA_WITH_AES_128_CBC_SHA

TLS_DHE_RSA_WITH_AES_256_CBC_SHA

TLS_DHE_DSS_WITH_AES_128_CBC_SHA

TLS_DHE_DSS_WITH_AES_256_CBC_SHA

SSL_RSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
```

```
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA

TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA

SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA

SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA

SSL_RSA_WITH_DES_CBC_SHA

SSL_DHE_RSA_WITH_DES_CBC_SHA

SSL_DHE_DSS_WITH_DES_CBC_SHA

SSL_RSA_EXPORT_WITH_DES40_CBC_SHA

SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA

SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

SSL_RSA_WITH_NULL_MD5

SSL_RSA_WITH_NULL_SHA

TLS_ECDH_ECDSA_WITH_NULL_SHA

TLS_ECDH_RSA_WITH_NULL_SHA

TLS_ECDHE_ECDSA_WITH_NULL_SHA

TLS_ECDHE_RSA_WITH_NULL_SHA

TLS_DH_anon_WITH_AES_128_CBC_SHA

TLS_DH_anon_WITH_AES_256_CBC_SHA

SSL_DH_anon_WITH_3DES_EDE_CBC_SHA

SSL_DH_anon_WITH_DES_CBC_SHA

TLS_ECDH_anon_WITH_AES_128_CBC_SHA

TLS_ECDH_anon_WITH_AES_256_CBC_SHA

TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA

SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA

TLS_ECDH_anon_WITH_NULL_SHA
```

```
TLS_KRB5_WITH_3DES_EDE_CBC_SHA

TLS_KRB5_WITH_3DES_EDE_CBC_MD5

TLS_KRB5_WITH_DES_CBC_SHA

TLS_KRB5_WITH_DES_CBC_MD5

TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA

TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5
```

21 Appendix F: Syntax and Format Reference

Regular Expression Syntax

Several transformation patterns allow using Regular Expressions to define the textual pattern we want to find in the host screen.

General Examples

Expression	Search for
Just some text	Specific text, find only "Just some text"
.*\.txt	Text files, like "Readme.txt"
Gr[ae]y	Only "Gray" or "Grey"
Colou?r	Only "Color" or "Colour"
\b[1-9][0-9]{2,4}\b	A number between 100 and 99999
\b[A-Z0-9._%-]+@ [A-Z0-9.-]+\.[A-Z]{2,4}\b	Email address, like "help@softwareag.com"

Host Specific Examples

Search for	Use...	Expression
"1 more >" or "2 more >"	To convert the text "1 more >" or "2 more >" etc. to a button that sends PF11	[1-9] more >
"(x-y)", "(January-march)", etc.	To erase any text with this pattern	\(.*-.*\)

General Rules

- [\|] separates alternatives.
- Expressions within parentheses are matched as subpattern groups and saved for further use.
- By default, a quantified subpattern matches as many times as possible without causing the rest of the pattern not to match. To change the quantifiers to match the minimum number of times possible, without causing the rest of the pattern not to match, use a [?] right after the quantifier.

Regular Expression Matching

Expression	Matches
{n,m}	At least n but not more than m times
{n,}	At least n times
{n}	Exactly n times
*	0 or more times
+	1 or more times
?	0 or 1 time
.	Everything except \n in a regular expression within parentheses
^	A null token matching the beginning of a string or line (i.e., the position right after a new line or right before the beginning of a string) in a regular expression within parentheses
$	A null token matching the end of a string or line (that is, the position right before a new line or right after the end of a string) in a regular expression within parentheses
\b	Backspace inside a character class ([abcd])
\b	Null token matching a word boundary (\w on one side and \W on the other)
\B	Null token matching a boundary that isn't a word boundary
\A	Only at the beginning of a string
\Z	Only at the end of a string (or before a new line at the end)
\	New line
\r	Carriage return
\t	Tab
\f	Form feed
\d	Digit [0-9]
\D	Non-digit [^0-9]
\w	Word character [0-9a-z_A-Z]
\W	Non-word character [^0-9a-z_A-Z]
\s	A white space character [\t\n\r\f]
\S	A non-white space character [^ \t\n\r\f]
\xnn	The hexadecimal representation of character nn
\cD	The corresponding control character
\nn or \nnn	The octal representation of character nn unless a back reference.
\1, \2, \3 ...	Whatever the first, second, third, and so on, parenthesized group matched. This is called a back reference. If there is no corresponding group, the number is interpreted as an octal representation of a character.
\0	The null character. Any other back-slashed character matches itself.
*?	0 or more times
+?	1 or more times

Expression	Matches
??	0 or 1 time
{n}?	Exactly n times
{n,}?	At least n times
{n,m}?	At least n but not more than m times

Date and Time Patterns

Date and time formats are specified by date and time pattern strings. Within date and time pattern strings, unquoted letters from 'A' to 'Z' and from 'a' to 'z' are interpreted as pattern letters representing the components of a date or time string. Text can be quoted using single quotes (') to avoid interpretation. Quotation marks ("''") represent a single quote. All other characters are not interpreted; they're simply copied into the output string during formatting or matched against the input string during parsing.

The following pattern letters are defined (all other characters from 'A' to 'Z' and from 'a' to 'z' are reserved):

Letter	Date or Time Component	Examples
G	Era designator	AD
y	Year	1996; 96
M	Month in year	July; Jul; 07
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day in week	Tuesday; Tue
a	Am/pm marker	PM
H	Hour in day (0-23)	0
k	Hour in day (1-24)	24
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978

Pattern letters are usually repeated, as their number determines the exact presentation:

- **Text**: For formatting, if the number of pattern letters is 4 or more, the full form is used; otherwise a short or abbreviated form is used if available. For parsing, both forms are accepted, independent of the number of pattern letters.
- **Number:** For formatting, the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount. For parsing, the number of pattern letters is ignored unless it is needed to separate two adjacent fields.
- **Year:** For formatting, if the number of pattern letters is 2, the year is truncated to 2 digits; otherwise it is interpreted as a number. For parsing, if the number of pattern letters is more than 2, the year is interpreted literally, regardless of the number of digits. So using the pattern "MM/dd/yyyy", "01/11/12" parses to Jan 11, 12 A.D.

  For parsing with the abbreviated year pattern ("y" or "yy"), SimpleDateFormat must interpret the abbreviated year relative to a century. It does this by adjusting dates to be within 80 years before and 20 years after the time the SimpleDateFormat instance is created. For example, using a pattern of "MM/dd/yy" and a SimpleDateFormat instance created on Jan 1, 1997, the string "01/11/12" would be interpreted as Jan 11, 2012 while the string "05/04/64" would be interpreted as May 4, 1964. During parsing, only strings consisting of exactly two digits will be parsed into the default century. Any other numeric string, such as a one digit string, a three or more digit string, or a two digit string that isn't all digits (for example, "-1"), is interpreted literally. So "01/02/3" or "01/02/003" are parsed, using the same pattern, as Jan 2, 3 AD. Likewise, "01/02/-3" is parsed as Jan 2, 4 BC.
- **Month:** If the number of pattern letters is 3 or more, the month is interpreted as text; otherwise, it is interpreted as a number.

Examples

The following examples show how date and time patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Date and Time Pattern	Result
yyyy-MM-dd	2006-12-31
dd/MMM/yy	31/Dec/06
mmddyy	123106
d.M.yy	31.12.06
"yyyy.MM.dd G 'at' HH:mm:ss "	2001.07.04 AD at 12:08:56
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o''clock' a,"	12 o'clock PM,
"K:mm a,"	0:08 PM,
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss"	Wed, 4 Jul 2001 12:08:56
"yyMMddHHmmss"	010704120856

Number Format

The number format expression is used for customizing the string representation of numbers. It can be used for adding grouping and decimal symbols, scientific notation, special prefixes and suffixes (such as currency signs or '%') and much more. The Format Number Expression allows the user to do so by defining a formatting pattern, a string written in a special syntax explained below, according to which the string representation of the number will be generated.

Several usage examples:

Input	Pattern	Resulting string
56	00000	00056
-56	#; (#)	(56)
234.34556	#0.00	234.35
4238476349587	#,##0	4,238,476,349,587
4238476349587	#,###0	4,2384,7634,9587
0.00000034545	0.#E0	3.5E-7
2347	#$	2347$

The format pattern contains a positive and negative subpattern, for example, "#,##0.00;(#,##0.00)". Each subpattern has a prefix, numeric part, and suffix. The negative subpattern is optional; if absent, then the positive subpattern prefixed with the localized minus sign (code>'-' for most languages) is used as the negative subpattern. That is, "0.00" alone is equivalent to "0.00;-0.00". If there is an explicit negative subpattern, it serves only to specify the negative prefix and suffix; the number of digits, minimal digits, and other characteristics are all the same as the positive pattern. That means that "#,##0.0#;(#)" produces precisely the same behavior as "#,##0.0#;(#,##0.0#)".

The prefixes, suffixes, and various symbols used for infinity, digits, thousands separators, decimal separators, etc. may be set to arbitrary values, and they will appear properly during formatting. However, care must be taken that the symbols and strings do not conflict, or parsing will be unreliable. For example: the decimal separator and thousands separator should be distinct characters, or parsing will be impossible.

The grouping separator is commonly used for thousands, but in some countries it separates ten-thousands. The grouping size is a constant number of digits between the grouping characters, such as 3 for 100,000,000 or 4 for 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. So "#,##,###,####" == "######,####" == "##,####,####".

Special Pattern Characters

Many characters in a pattern are taken literally; they are matched during parsing and output unchanged during formatting. Special characters, on the other hand, stand for other characters, strings, or classes of characters. They must be quoted, unless noted otherwise, if they are to appear in the prefix or suffix as literals.

Symbol	Location	Localized?	Meaning
0	Number	Yes	Digit
#	Number	Yes	Digit, zero shows as absent
.	Number	Yes	Decimal separator or monetary decimal separator
-	Number	Yes	Minus sign
,	Number	Yes	Grouping separator
E	Number	Yes	Separates mantissa and exponent in scientific notation. Need not be quoted in prefix or suffix.
;	Subpattern boundary	Yes	Separates positive and negative subpatterns
%	Prefix or suffix	Yes	Multiply by 100 and show as percentage
\u2030	Prefix or suffix	Yes	Multiply by 1000 and show as per mille
¤ (\u00A4)	Prefix or suffix	No	Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.
'	Prefix or suffix	No	Used to quote special characters in a prefix or suffix, for example, "'#'#" formats 123 to "#123". To create a single quote itself, use two in a row: "# o''clock".

Number Format in Different Languages

Although the decimal and the grouping separators used in patterns must be '.' and ',' respectively, they might be replaced with different separators during the formatting process, depending on the application language. For example: using the pattern "#,##0.00" to format the number 18734573.07 will yield the string "18,734,573.07" if the application language is English, and "18.734.573,07" in case the application language is Italian. The default minus sign (normally '-') is also determined according to the application language.

Scientific Notation

Numbers in scientific notation are expressed as the product of a mantissa and a power of ten, for example, 1234 can be expressed as 1.234 x 10^3. The mantissa is often in the range 1.0 <= x < 10.0, but it need not be. In a pattern, the exponent character immediately followed by one or more digit characters indicates scientific notation. Example: "0.###E0" formats the number 1234 as "1.234E3".

- The number of digit characters after the exponent character gives the minimum exponent digit count. There is no maximum. Negative exponents are formatted using the localized minus sign, not the prefix and suffix from the pattern. This allows patterns such as "0.###E0 m/s".
- The minimum and maximum number of integer digits are interpreted together:

- If the maximum number of integer digits is greater than their minimum number and greater than 1, it forces the exponent to be a multiple of the maximum number of integer digits, and the minimum number of integer digits to be interpreted as 1. The most common use of this is to generate engineering notation, in which the exponent is a multiple of three, e.g., "##0.#####E0". Using this pattern, the number 12345 formats to "12.345E3", and 123456 formats to "123.456E3".
- Otherwise, the minimum number of integer digits is achieved by adjusting the exponent. Example: 0.00123 formatted with "00.###E0" yields "12.3E-4".

- The number of significant digits in the mantissa is the sum of the minimum integer and maximum fraction digits, and is unaffected by the maximum integer digits. For example, 12345 formatted with "##0.##E0" is "12.3E3". To show all digits, set the significant digits count to zero. The number of significant digits does not affect parsing.
- Exponential patterns may not contain grouping separators.

Rounding

The number format uses half-even rounding for formatting (round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor).

Format Number Formal Syntax

Pattern
PositivePattern

PositivePattern ; NegativePattern

PositivePattern
Prefix$_{opt}$ Number Suffix$_{opt}$

NegativePattern
Prefix$_{opt}$ Number Suffix$_{opt}$

Prefix
any Unicode characters except \uFFFE, \uFFFF, and special characters

Suffix
Unicode characters except \uFFFE, \uFFFF, and special characters

Number
Integer Exponent$_{opt}$

Integer . Fraction Exponent$_{opt}$

Integer
MinimumInteger

# Integer

# , Integer

MinimumInteger

0 MinimumInteger

0 , MinimumInteger

Fraction

MinimumFraction$_{opt}$ OptionalFraction$_{opt}$

OptionalFraction

# OptionalFraction$_{opt}$

Exponent

E MinimumExponent

MinimumExponent

0 MinimumExponent$_{opt}$

22 Appendix G: ASCII Character Table

Decimal	Octal	Hex	Binary	Value	
000	000	000	00000000	NUL	(Null char.)
001	001	001	00000001	SOH	(Start of Header)
002	002	002	00000010	STX	(Start of Text)
003	003	003	00000011	ETX	(End of Text)
004	004	004	00000100	EOT	(End of Transmission)
005	005	005	00000101	ENQ	(Enquiry)
006	006	006	00000110	ACK	(Acknowledgment)
007	007	007	00000111	BEL	(Bell)
008	010	008	00001000	BS	(Backspace)
009	011	009	00001001	HT	(Horizontal Tab)
010	012	00A	00001010	LF	(Line Feed)
011	013	00B	00001011	VT	(Vertical Tab)
012	014	00C	00001100	FF	(Form Feed)
013	015	00D	00001101	CR	(Carriage Return)
014	016	00E	00001110	SO	(Shift Out)
015	017	00F	00001111	SI	(Shift In)
016	020	010	00010000	DLE	(Data Link Escape)
017	021	011	00010001	DC1 (XON) (Device Control 1)	
018	022	012	00010010	DC2	(Device Control 2)
019	023	013	00010011	DC3 (XOFF)(Device Control 3)	
020	024	014	00010100	DC4	(Device Control 4)

Decimal	Octal	Hex	Binary	Value	
021	025	015	00010101	NAK	(Negative Acknowledgment)
022	026	016	00010110	SYN	(Synchronous Idle)
023	027	017	00010111	ETB	(End of Trans. Block)
024	030	018	00011000	CAN	(Cancel)
025	031	019	00011001	EM	(End of Medium)
026	032	01A	00011010	SUB	(Substitute)
027	033	01B	00011011	ESC	(Escape)
028	034	01C	00011100	FS	(File Separator)
029	035	01D	00011101	GS	(Group Separator)
030	036	01E	00011110	RS	(Request to Send)(Record Separator)
031	037	01F	00011111	US	(Unit Separator)
032	040	020	00100000	SP	(Space)
033	041	021	00100001	!	(exclamation mark)
034	042	022	00100010	"	(double quote)
035	043	023	00100011	#	(number sign)
036	044	024	00100100	$	(dollar sign)
037	045	025	00100101	%	(percent)
038	046	026	00100110	&	(ampersand)
039	047	027	00100111	'	(single quote)
040	050	028	00101000	(	(left/opening parenthesis)
041	051	029	00101001	)	(right/closing parenthesis)
042	052	02A	00101010	*	(asterisk)
043	053	02B	00101011	+	(plus)
044	054	02C	00101100	,	(comma)
045	055	02D	00101101	-	(minus or dash)
046	056	02E	00101110	.	(dot)
047	057	02F	00101111	/	(forward slash)
048	060	030	00110000	0	
049	061	031	00110001	1	
050	062	032	00110010	2	
051	063	033	00110011	3	
052	064	034	00110100	4	
053	065	035	00110101	5	
054	066	036	00110110	6	
055	067	037	00110111	7	
056	070	038	00111000	8	

Decimal	Octal	Hex	Binary	Value	
057	071	039	00111001	9	
058	072	03A	00111010	:	(colon)
059	073	03B	00111011	;	(semi-colon)
060	074	03C	00111100	<	(less than)
061	075	03D	00111101	=	(equal sign)
062	076	03E	00111110	>	(greater than)
063	077	03F	00111111	?	(question mark)
064	100	040	01000000	@	(AT symbol)
065	101	041	01000001	A	
066	102	042	01000010	B	
067	103	043	01000011	C	
068	104	044	01000100	D	
069	105	045	01000101	E	
070	106	046	01000110	F	
071	107	047	01000111	G	
072	110	048	01001000	H	
073	111	049	01001001	I	
074	112	04A	01001010	J	
075	113	04B	01001011	K	
076	114	04C	01001100	L	
077	115	04D	01001101	M	
078	116	04E	01001110	N	
079	117	04F	01001111	O	
080	120	050	01010000	P	
081	121	051	01010001	Q	
082	122	052	01010010	R	
083	123	053	01010011	S	
084	124	054	01010100	T	
085	125	055	01010101	U	
086	126	056	01010110	V	
087	127	057	01010111	W	
088	130	058	01011000	X	
089	131	059	01011001	Y	
090	132	05A	01011010	Z	
091	133	05B	01011011	[	(left/opening bracket)
092	134	05C	01011100	\	(back slash)

Decimal	Octal	Hex	Binary	Value	
093	135	05D	01011101	]	(right/closing bracket)
094	136	05E	01011110	^	(caret/cirumflex)
095	137	05F	01011111	_	(underscore)
096	140	060	01100000	`	
097	141	061	01100001	a	
098	142	062	01100010	b	
099	143	063	01100011	c	
100	144	064	01100100	d	
101	145	065	01100101	e	
102	146	066	01100110	f	
103	147	067	01100111	g	
104	150	068	01101000	h	
105	151	069	01101001	I	
106	152	06A	01101010	j	
107	153	06B	01101011	k	
108	154	06C	01101100	l	
109	155	06D	01101101	m	
110	156	06E	01101110	n	
111	157	06F	01101111	o	
112	160	070	01110000	p	
113	161	071	01110001	q	
114	162	072	01110010	r	
115	163	073	01110011	s	
116	164	074	01110100	t	
117	165	075	01110101	u	
118	166	076	01110110	v	
119	167	077	01110111	w	
120	170	078	01111000	x	
121	171	079	01111001	y	
122	172	07A	01111010	z	
123	173	07B	01111011	{	(left/opening brace)
124	174	07C	01111100	|	(vertical bar)
125	175	07D	01111101	}	(right/closing brace)
126	176	07E	01111110	~	(tilde)
127	177	07F	01111111	DEL	(delete)

23 Appendix H: SDFX File Format Definition

Introduction

SDFX (Screen Definition Format) is a proprietary generic format used for describing a host screen. In addition to standard formats like BMS, MFS and NATURAL maps, ApplinX supports this generic format. If your host screen maps are not supported in ApplinX you can convert them to the SDFX format and then import them into your ApplinX repository.

SDFX is based on the XML structure. Each SDFX file is composed of one mapset, and must have the ".SDFX" extension. A mapset contains one or more smaller units called maps. Each map currently corresponds to one host screen. A map is made up of identifiers, fields, and map steps. Mapsets, maps, and fields are identified by name. ApplinX uses the map name to create a screen entity with the same name.

Mapset File Structure

The following details the mapset file structure, where all the map and field details are placed within the Map tag and are detailed in **Defining Maps** and **Defining Fields**.

```
<Mapset>
    <Name>mapsetname</Name>
    <Version>1</Version>
    <Type>SIMPLE</Type>
    <Map>
       ...
    <Map>
    <Map>
       ...
    <Map>
<Mapset>
```

Mapset Parameters

`<Name>mapsetname</Name>`
The name of the mapset (mandatory).

`<Version>1</Version>`
This line contains the SDFX format version (currently, always 1).

`<Type>SIMPLE</Type>`
This line contains the type of SDFX used. Currently the only type available is "SIMPLE".

`<Map> ... <Map>`
Refer to **Defining Maps**. A number of maps can be added to the same file.

Defining Maps

Each map represents a single screen and contains the screen properties such as the screen name, screen position, screen width and height etc.

```
<Map>
    <Name>MapName</Name>
    <PosX>1</PosX>
    <PosY>1</PosY>
    <Width>80</Width>
    <Height>24</Height>
    <IsPopup>false</IsPopup>
    <Field class="SimpleField">
      ...
    </Field>
    <Field class="ListField">
      ...
    </Field>
    <LoopField>
      ...
    </LoopField>
</Map>
```

Map Parameters

`<Name>MapName</Name>`
: The name of the map (mandatory).

`<PosX>1</PosX>`
: The map's horizontal position (the X coordinate of the rectangle's top-left character, can be from 1 to the width of the screen). When not specified, the default is 1.

`<PosY>1</PosY>`
: The map's vertical position (the Y coordinate of the rectangle's top-left character, can be from 1 to the height of the screen). When not specified, the default is 1.

`<Width>80</Width>`
: The map's width (optional, default is 80).

`<Height>24</Height>`
: The map's height (optional, default is 24).

`<IsPopup>false</IsPopup>`
: True indicates that this map is used as a pop-up on the host (optional, default is false).

`<Field class="SimpleField"> ... </Field>`
: Refer to **Defining Fields**. A number of fields can be added to the same map.

Defining Identifiers

Use SDFX identifiers to create ApplinX identifiers (refer to Defining Identifiers).

Parameters to use for a Text identifier

The identifier is recognized if its text matches (or does not match) the host screen text in the defined position.

`Text`
: The string to try and match.

`IsMatch`
: Boolean (true \ false), default value is true. When set to false, then the identifier is recognized only if the host screen's text \ attribute in that position does not match the identifier's text \ attribute.

`IsCaseSensitive`
: Boolean (true \ false). When set to true, then the text will be case sensitive.

`ScreenArea`
: Represents the screen area within in which to search for the identifier. Possible values:

  `Anywhere`
  : Any position on the screen is a valid position.

    For example

```
<ScreenArea class="Anywhere"/>
```

  `Position`
  : Defines a position on the screen. `StartPos` determines the first position within the screen.

    For example:

```
<ScreenArea class="Position">
   <StartPos>
     <PosX>40</PosX>
     <PosY>23</PosY>
   </StartPos>
 </ScreenArea>
```

  `PositionLength`
  : Defines the first position within the screen and the length of the area.

    For example:

```
<ScreenArea class="PositionLength">
  <StartPos>
    <PosX>40</PosX>
    <PosY>23</PosY>
  </StartPos>
  <Length>15</Length>
</ScreenArea>
```

`Rectangle`

Defines a rectangle on the screen. `StartPos` determines the top left position and `EndPos` determines the bottom right position. Together these create a rectangle.

For example:

```
<ScreenArea class="Rectangle">
  <StartPos>
    <PosX>10</PosX>
    <PosY>1</PosY>
  </StartPos>
  <EndPos>
    <PosX>30</PosX>
    <PosY>24</PosY>
  </EndPos>
</ScreenArea>
```

The following example is sample code used for a text identifier, using the type "Anywhere".

```
<Identifier class="Text">
      <Text>TextToMatch</Text>
      <IsMatch>true</IsMatch>
      <IsCaseSensitive>true</IsCaseSensitive>
      <ScreenArea class="Anywhere"/>
 </Identifier>
```

Parameters to use for an Attribute identifier

The identifier is recognized if its attribute matches the attribute that applies for the host screen in the defined position. The position does not necessarily has to be the position of the attribute character.

`Attribute`

The attribute to search for. Possible values: PROTECTED, HIDDEN, INTENSIFIED, and REVERSED_VIDEO.

`ScreenArea: Position`

Defines a position on the screen where the attribute should be checked.

For example:

```
<ScreenArea class="Position">
   <StartPos>
     <PosX>40</PosX>
     <PosY>23</PosY>
   </StartPos>
 </ScreenArea>
```

The following example is sample code used for an attribute identifier (defines an "Unprotected" attribute).

```
<Identifier class="Attribute">
     <Attribute>PROTECTED</Attribute>
     <IsMatch>false</IsMatch>
     <ScreenArea class="Position">
       <StartPos>
         <PosX>40</PosX>
         <PosY>23</PosY>
       </StartPos>
     </ScreenArea>
 </Identifier>
```

Defining Fields

- **Definitions of a Simple Field**: Simple fields can be protected, unprotected or static fields.
- **Definitions of a List Field**: A List Field is a field which contains a repetitive pattern of simple fields (a "table" of similar cells).
- **Definition of a Loop Field**: A loop field describes a repetitive element of the host screen, which is not a list.

Definitions of a Simple Field

A simple field may be a protected, unprotected or status field. A field for which no name is given, will be considered by ApplinX to be an identifier. A simple field can also be used as a static identifier.

```
  <Field class="SimpleField">
      <Type>SIMPLE</Type>
      <Name>DFHM001</Name>
      <PosX>2</PosX>
      <PosY>1</PosY>
      <Length>4</Length>
      <Text>string</Text>
      <Attribute>PROTECTED</Attribute>
      <DataType>DATA_TYPE_ALPHANUMERIC</DataType>
      <Bright>NORM</Bright>
      <ForegroundColor>WHITE</ForegroundColor>
      <BackgroundColor>DEFAULT</BackgroundColor>
      <HighlightBlink>false</HighlightBlink>
      <HighlightReverse>false</HighlightReverse>
      <HighlightUnderline>false</HighlightUnderline>

  </Field>
```

Note: A field which does not have a name tag but has a text tag, will be used as a screen identifier.

Simple Field Parameters

`<Field class="SimpleField">`
: Indicates that this field is a simple field (a field's class can be SimpleField or ListField).

`<Type>SIMPLE</Type>`
: Indicates that it is a simple field (optional, since SIMPLE is the default field type).

`<Name>DFHM001</Name>`
: The name of the field (optional). A field for which no name is given will be considered by Applinx to be an identifier (a static area on the screen).

`<PosX>2</PosX>`
: The field's horizontal position (the X coordinate of the position of the attribute character, can be from 1 to the width of the screen), relative to the original map.

`<PosY>1</PosY>`
: The field's vertical position (the Y coordinate of the position of the attribute character, can be from 1 to the height of the screen), relative to the original map.

`<Length>4</Length>`
: The field's length (not including the attribute character).

`<Text>string</Text>`
: The initial value of the field (optional). This should match the field's length and will be truncated or space-padded as needed.

`<Attribute>PROTECTED</Attribute>`
: The field's attribute. Possible values are: "UNPROTECTED" or "PROTECTED". The default value is "PROTECTED".

`<DataType>DATA_TYPE_ALPHANUMERIC</DataType>`
: Determines whether the field can receive numeric or alpha numeric data. Possible values: "DATA_TYPE_ALPHANUMERIC" or "DATA_TYPE_NUMERIC". The default value is "DATA_TYPE_ALPHANUMERIC".

`<Bright>NORM</Bright>`
: The field's attribute (brightness) (optional). Possible values: "BRT", "NORM" or "DRK". The default value is "NORM".

`<ForegroundColor>WHITE</ForegroundColor>`
Indicates the foreground color of the field (optional). The default color is white. The color must be written in capital letters. Possible values include: WHITE, BLACK, AQUA, BLUE, BROWN, GRAY, GREEN, LIGHT_AQUA, LIGHT_BLUE, LIGHT_GREEN, LIGHT_PURPLE, LIGHT_RED, LIGHT_WHITE, PURPLE, RED, YELLOW, NONE, DEFAULT.

`<BackgroundColor>DEFAULT</BackgroundColor>`
Indicates the foreground color of the field (optional). The default color is black. The color must be written in capital letters. Possible values include: WHITE, BLACK, AQUA, BLUE, BROWN_YELLOW, GREEN, PURPLE, RED, NONE, DEFAULT.

`<HighlightBlink>false</HighlightBlink>`
BLINK highlighting (optional, default is false).

`<HighlightReverse>false</HighlightReverse>`
REVERSE highlighting (optional, default is false).

`<HighlightUnderline>false</HighlightUnderline>`
UNDERLINE highlighting (optional, default is false).

Definitions of a List Field

A List Field is a field which contains a repetitive pattern of simple fields. Since this repetition can expand vertically to the right (not only downward), a list can be seen as a "table" of similar cells (Applinx multiple fields \ tables). Each cell in the list is called a record and can contain several fields. You are required to define the first record (cell) and it must consist of Simple fields only.

```
<Field class="ListField">
      <Type>LIST</Type>
      <Name>DFHM002</Name>
      <PosX>1</PosX>
      <PosY>1</PosY>
      <Direction>HORIZONTAL_FIRST</Direction>
      <RecordWidth>1</RecordWidth>
      <RecordHeight>1</RecordHeight>
      <RecordsNum>4</RecordsNum>
      <Field>
       ...
      </Field>
      <Field>
       ...
      </Field>
</Field>
```

List Field Parameters

`<Field class="ListField">`
Indicates that this field is a list field (a field's class can be SimpleField or ListField).

`<Type>LIST</Type>`
: Indicates that it is a list field.

`<Name>DFHM002</Name>`
: The name of the list (optional).

`<PosX>1</PosX>`
: The list's horizontal position (the X coordinate of its top-left corner, can be from 1 to the width of the screen). Note that the corner of the list's area isn't necessarily the corner of the top-left cell (unless the cell's position is 1,1).

`<PosY>1</PosY>`
: The list's vertical position (the Y coordinate of its top-left corner, can be from 1 to the height of the screen).

`<Direction>HORIZONTAL_FIRST</Direction>`
: The direction in which records are inserted into the list (optional). Possible values are: "VERTICAL_FIRST" or "HORIZONTAL_FIRST" (default: VERTICAL_FIRST).

`<RecordWidth>1</RecordWidth>`
: The width (in characters) of a single list record (i.e. cell). The default value is "1". For example, when wanting to repeat an occurrence every twenty characters (counting from the beginning of the previous record), enter 20 for this parameter and 0 for the value of the RecordHeight.

`<RecordHeight>1</RecordHeight>;`
: The height (in characters) of a single list record (i.e. cell). Possible values can be in the range 1-99. The default value is "1". For example, when wanting to repeat an occurrence every two rows (counting from the first row of the previous occurrence), enter 2 for this parameter.

`<RecordsNum>4</RecordsNum>`
: The total number of records to produce. Possible values can be in the range 1-99. The default value is "1".

Definition of a Loop Field

A loop field describes a repetitive element of the host screen, which is not a list. The most common example being the drawing of a frame of some sort which consists of simple (static) fields one below the other. Each simple field inside the loop field is repeated (cloned) either along the x-axis, y-axis or both ("table"). The distance between adjacent cloned fields is controllable, and it is the user's responsibility to make sure cloned fields are kept within the map's bounds.

```
<LoopField>
   <HorizontalClonesNum>1</HorizontalClonesNum>
   <HorizontalOffset>1</HorizontalOffset>
   <VerticalClonesNum>1</VerticalClonesNum>
   <VerticalOffset>1</VerticalOffset>
   <Field>
     ...
   </Field>
   <Field>
```

```
      ...
    </Field>
 </LoopField>
```

Loop Field Parameters

`<HorizontalClonesNum>1</HorizontalClonesNum>`
: The number of cloned fields (including the field itself) to be created in each row (optional, default is 1).

`<HorizontalOffset>1</HorizontalOffset>`
: The number of columns between adjacent cloned fields (optional, default is 1 which means that there are no blank columns between cloned fields, just attributes).

`<VerticalClonesNum>1</VerticalClonesNum>`
: The number of cloned fields (including the field itself) to be created in each column (optional, default is 1).

`<VerticalOffset>1</VerticalOffset>`
: The number of rows between adjacent cloned fields (optional, default is 1 which means that there are no blank rows between cloned fields).

Defining Map Steps

The Map Steps defined here can be used in the Application Map.

```
<MapStep>
   <SendKeys>[ENTER]</SendKeys>
   <TargetScreen>ScreenName</TargetScreen>
   <CursorPos>
     <PosX>10</PosX>
     <PosY>5</PosY>
   </CursorPos>
   <InputField>
     <StartPos>
       <PosX>10</PosX>
       <PosY>5</PosY>
     </StartPos>
     <Text>input</Text>
   </InputField>
   <InputField>
     <StartPos>
       <PosX>10</PosX>
       <PosY>7</PosY>
     </StartPos>
     <GlobalVariableName>variableName</GlobalVariableName>
     <Text>prefix</Text>
   </InputField>
   <InputField>
```

```
      ...
    </InputField>
    <InputField>
      ...
    </InputField>
  </Mapstep>
```

Note:

1. It is only possible to place one map step from one specific screen to another specific target screen. When defining more than one, only the first one defined in the file will be displayed (e.g. if you can get from screen A to screen B both by sending the [enter] key or by sending the [PF5] key, the first map step defined is the one ApplinX uses).
2. It is not possible to define a map step from a screen to itself. Such map steps, if defined in the SDFX file, will be ignored.
3. Map step must contain a "SendKeys" element and a "TargetScreen" element. If one of them is missing, then the map step will not be added (a warning will be added to the server log).

Map Step Parameters

`SendKeys`

The host key. Ensure that the host key you define is listed in the list of host keys which is displayed in the application properties, in the Keyboard Mapping tab, in the Host Key column.

For Example:

```
<SendKeys>[attn]</SendKeys>
```

`CursorPos`

The position of the cursor. For example there may be several map steps with the same [help] host key, each with a different target screen (the specific help menu) as long as they also have different cursor locations (the field for which the user wanted the help menu).

`PosX`

The column of the cursor position.

`PosY`

The row of the cursor position.

`TargetScreen`

The name of the destination screen, as defined in the destination screen's map definition in the same SDFX file.

`InputField`

Optional. Each map step can have zero or more input fields, each field represents input the user entered to a specific field when performing the map step.

`StartPos`

Determines the position on screen where the first character was entered.

`PosX`
The column of the cursor position.

`PosY`
The row of the cursor position.

`GlobalVariableName`
Optional. The name of the global variable whose value should be used as the text.

`Text`
The text entered.

Example of an SDFX File

```
<Mapset>
  <Name>Example</Name>
  <Version>1</Version>
  <Type>SIMPLE</Type>
  <Map>
    <Name>Example</Name>
    <PosX>1</PosX>
    <PosY>1</PosY>
    <Width>80</Width>
    <Height>24</Height>
    <CursorPosX>1</CursorPosX>
    <CursorPosY>1</CursorPosY>
    <IsPopup>false</IsPopup>
 <Field class="SimpleField">
      <Length>10</Length>
      <Attribute>PROTECTED</Attribute>
      <DataType>DATA_TYPE_ALPHANUMERIC</DataType>
      <Bright>NORM</Bright>
      <ForegroundColor>GREEN</ForegroundColor>
      <BackgroundColor>DEFAULT</BackgroundColor>
      <DbcsMode>NONE</DbcsMode>
      <HighlightBlink>false</HighlightBlink>
      <HighlightReverse>false</HighlightReverse>
      <HighlightUnderline>false</HighlightUnderline>
      <OffsetFromLeft>0</OffsetFromLeft>
      <OffsetFromRight>0</OffsetFromRight>
      <OffsetFromTop>0</OffsetFromTop>
      <OffsetFromBottom>0</OffsetFromBottom>
      <Type>SIMPLE</Type>
   <Text>First Name</Text>
      <PosX>8</PosX>
      <PosY>3</PosY>
    </Field>
 <Field class="SimpleField">
      <Length>10</Length>
      <Attribute>PROTECTED</Attribute>
      <DataType>DATA_TYPE_ALPHANUMERIC</DataType>
      <Bright>NORM</Bright>
      <ForegroundColor>WHITE</ForegroundColor>
      <BackgroundColor>DEFAULT</BackgroundColor>
      <Type>SIMPLE</Type>
   <Text>----------</Text>
      <PosX>8</PosX>
      <PosY>4</PosY>
    </Field>
 <Field class="SimpleField">
      <Length>10</Length>
      <Attribute>PROTECTED</Attribute>
      <DataType>DATA_TYPE_ALPHANUMERIC</DataType>
      <Bright>NORM</Bright>
      <ForegroundColor>WHITE</ForegroundColor>
      <BackgroundColor>DEFAULT</BackgroundColor>
      <OffsetFromLeft>0</OffsetFromLeft>
      <OffsetFromRight>0</OffsetFromRight>
      <OffsetFromTop>0</OffsetFromTop>
      <OffsetFromBottom>0</OffsetFromBottom>
      <Type>SIMPLE</Type>
   <Text>----------</Text>
      <PosX>20</PosX>
      <PosY>4</PosY>
    </Field>
 <Field class="SimpleField">
      <Length>10</Length>
      <Attribute>PROTECTED</Attribute>
      <DataType>DATA_TYPE_ALPHANUMERIC</DataType>
      <Bright>NORM</Bright>
      <ForegroundColor>BLUE</ForegroundColor>
      <BackgroundColor>DEFAULT</BackgroundColor>
      <DbcsMode>NONE</DbcsMode>
      <HighlightBlink>false</HighlightBlink>
      <HighlightReverse>false</HighlightReverse>
      <HighlightUnderline>false</HighlightUnderline>
      <OffsetFromLeft>0</OffsetFromLeft>
      <OffsetFromRight>0</OffsetFromRight>
      <OffsetFromTop>0</OffsetFromTop>
      <OffsetFromBottom>0</OffsetFromBottom>
      <Type>SIMPLE</Type>
   <Text>Last Name</Text>
      <PosX>20</PosX>
      <PosY>3</PosY>
    </Field>
    <Field class="ListField">
      <Direction>HORIZONTAL_FIRST</Direction>
      <RecordWidth>1</RecordWidth>
      <RecordHeight>1</RecordHeight>
      <RecordsNum>15</RecordsNum>
      <Field>
```

```
     <Name>firstName</Name>
        <Length>10</Length>
        <Attribute>PROTECTED</Attribute>
        <DataType>DATA_TYPE_ALPHANUMERIC</DataType>
        <Bright>BRT</Bright>
        <DbcsMode>NONE</DbcsMode>
        <Type>SIMPLE</Type>
        <PosX>8</PosX>
        <PosY>6</PosY>
      </Field>
      <Field>
     <Name>lastName</Name>
        <Length>10</Length>
        <Attribute>UNPROTECTED</Attribute>
        <DataType>DATA_TYPE_NUMERIC</DataType>
        <Bright>NORM</Bright>
        <ForegroundColor>WHITE</ForegroundColor>
        <BackgroundColor>DEFAULT</BackgroundColor>
        <Type>SIMPLE</Type>
        <PosX>20</PosX>
        <PosY>6</PosY>
      </Field>
      <Type>LIST</Type>
      <PosX>1</PosX>
      <PosY>1</PosY>
    </Field>
    <LoopField>
      <HorizontalClonesNum>1</HorizontalClonesNum>
      <HorizontalOffset>1</HorizontalOffset>
      <VerticalClonesNum>1</VerticalClonesNum>
      <VerticalOffset>1</VerticalOffset>
    </LoopField>
  </Map>
</Mapset>
```

24 Appendix I: Dynamic Field Mapping Limitations

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 Map Fields section for further details as to how to map application fields according to their leading label.

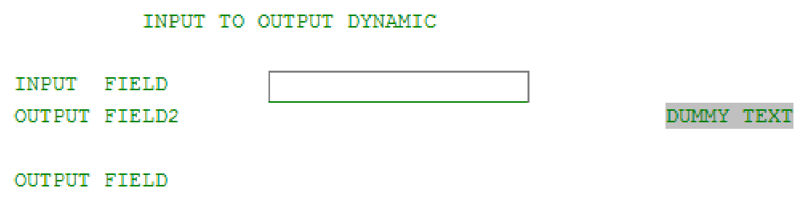
Following are some limitations:

There is no specific identifier on the host screen which distinguishes between protected fields and static text which cannot change in runtime. This implies the following limitations:

- If a field does not have a distinctive label exactly to its left, it cannot be defined as a dynamic field.
- This feature is disabled for applications that are defined with a right to left language (a new dynamic field mapping can't be created).
- This feature is disabled, when working with a character based host (VT).
- Static and Dynamic field mappings should not be set to match the same fields.

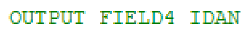
Specific use cases

- When the protected field is blank, any text to the right of the field is matched as the protected field area.



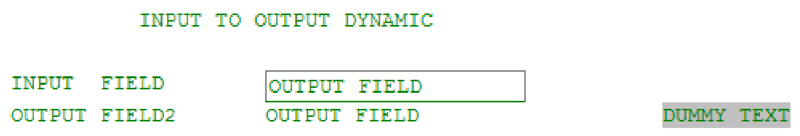
```
            INPUT TO OUTPUT DYNAMIC

    INPUT  FIELD         [                ]
    OUTPUT FIELD2                                    DUMMY TEXT

    OUTPUT FIELD
```

- When a label and field are separated by a single white space, the protected field won't be matched, unless the "field search area" and "label search area" do not overlap.



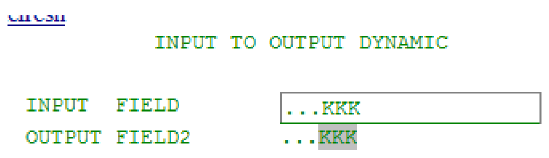
```
OUTPUT FIELD4 IDAN
```

- When arbitrary text in a protected field, matches the label search criteria it is wrongly matched as a field label, and the text to the right of it may be wrongly marked as a protected field.



```
            INPUT TO OUTPUT DYNAMIC

    INPUT  FIELD         [OUTPUT FIELD    ]
    OUTPUT FIELD2         OUTPUT FIELD                DUMMY TEXT
```

- When the text of the protected field starts with white spaces or separator characters, such as a dot or colon this text won't be considered as part of the field.



```
            INPUT TO OUTPUT DYNAMIC

    INPUT  FIELD         [...KKK          ]
    OUTPUT FIELD2         ...KKK
```

- If two white spaces or separator characters appear as a sequence inside a protected field, they will be considered as a field boundary. For example, a row part containing the text "Hello, world" will be identified as a protected field with the text "Hello" only. In addition, when a leading label is defined as a part of a protected field's actual label and the actual label contains two or more separator characters (after the given leading label), the rest of the actual label might be falsely marked as the protected field. Note that these limitations are data dependant, they may only surface when specific data is sent from the host and therefore may not be identified during desgin time but only in runtime.

25 Appendix J: Character Mode Hosts (VT Protocol)

Introduction

Character mode hosts are supported in SOA applications only. Character mode hosts are character-stream based, which means that the host sends buffers all the time and unlike the situation with block mode hosts, there is no way to divide these buffers to screens. The Session View can work both in character mode and in block mode. You will need to start working in character mode (the default mode) - in that mode the Session View acts like a regular emulator: each key is sent to the host. By using the character mode, you can navigate through screens.

In character mode based hosts, the whole screen received from the host is defined as unprotected. ApplinX is unable to determine where there are fields within the screen; therefore ApplinX needs to be taught where the fields are. To do this, create screens, define and map application fields. After you mapped the necessary application fields for a screen, you can transfer to block mode in the Session view (Refer to Navigating within a Session: Character Mode (VT) Hosts). Using Block mode means that you can type data in all the fields, but the data will not be sent to the host until a submit key such as ENTER or a PF key is pressed. The behavior in Path Procedures will be similar: in the Step node instructions for the whole screen are provided, but ApplinX sends the data character by character.

As in VT the host continuously sends data. When sending a submit key such as ENTER or a PF key, the host immediately returns the same screen that was sent (the correct screen content will only be displayed if the Synchronize with host icon is pressed), and only afterwards the actual screen is received. Therefore, in Path Procedures it is necessary to use Wait Conditions to define to wait for the screen to arrive from the host. Refer to Path Procedures to implement Wait Conditions. You may need to use different types of Wait Conditions for different use cases. Refer to Wait Conditions for further details.

Creating a Character Mode Host Based Application

Follow the step-by-step instructions for creating a new application.

VT Application Parameters

Host Related Parameters

Overriding the Application Parameters in a Specific Screen

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.

Screen Editor: VT Parameters tab

Overriding the Application Parameters in a Specific Field/ Setting the Behavior of Specific Field Types

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.

Fields tab: VT Parameters section

Navigating within a Session

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 the 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.

Wait Conditions in Path Procedures

As in character mode hosts, each key is submitted to the host. The host returns the same screen and immediately after sends the updated screen. This being so, a Wait Condition is required in order to wait for the following screen.

26 Appendix K: Host Supported Code Pages

Code	Language
037	US, Belgium, Brazil, Canada, Netherlands, Portugal
273	Germany, Austria
274	Belgium Old
275	Brazil Old
277	Denmark, Norway
278	Finland, Sweden
280	Italy
284	Spain, Latin America
285	UK
297	France
420	Arabic
424	Hebrew New Code
500	Multilingual
803	Hebrew Old Code
838	Thai
870	Latin 2 (EBCDIC Multilingual)
875	Greece
905	Turkish-Extended Code Page
930	Japan Katakana Ex
933	Korean
935	Simplified Chinese
937	Traditional Chinese
939	Japan English Ex
1025	1025 - Cyrillic

Code	Language
1026	Turkish
1130	Vietnamese
1155	Turkish - Euro support
1097	Farsi
1500	Multilingual Swiss
VT configurations	
SJIS	Japanese SJIS
EUC	Japanese EUC
874	Thai
916	Hebrew
932	Japanese ISO
1011	German
1024	Portuguese
1090	Decimal Special Graphics
1100	Multi
1134	Hebrew 7-bit
3000	Line Drawing
4001	International
4002	Multilingual
4040	Session 3151 Initial
4041	Session 3151
4090	ANSI Special Graphics
8000	ISO-8859-1