

IBM® Security Access Manager for Enterprise Single
Sign-On
Version 8.2.2

AccessStudio Guide



IBM® Security Access Manager for Enterprise Single
Sign-On
Version 8.2.2

AccessStudio Guide



Note

Before using this information and the product it supports, read the information in "Notices" on page 165.

Edition notice

Note: This edition applies to version 8.2.2 of IBM Security Access Manager for Enterprise Single Sign-On, (product number 5724-V67) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2002, 2015.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	v
--------------------------	----------

About this publication	vii
---	------------

Accessibility	vii
Technical training	vii
Support information	vii
Statement of Good Security Practices.	vii

Chapter 1. About AccessStudio	1
--	----------

AccessStudio features and benefits	1
How AccessStudio works	2
AccessStudio basic concepts	3
Advanced concepts	5
AccessStudio interface	8
Menu bar	8
Toolbar	9
Data type pane	9
Details pane	10
Message pane	13

Chapter 2. AccessProfiles	15
--	-----------

Creating AccessProfiles in AccessStudio	15
---	----

Chapter 3. Standard AccessProfiles	17
---	-----------

Creating standard AccessProfiles for different application types	17
Creating AccessProfiles for Windows applications	17
Creating AccessProfiles for Web applications	23
Creating logon AccessProfiles for Java applications	28
Creating logon AccessProfiles for applications or applets that use Java	30
Creating AccessProfiles for terminal applications	31
Creating AccessProfiles for Mainframe or cursor-based applications.	33
Creating logon AccessProfiles for Mainframe applications with HLLAPI support	38
Creating logon AccessProfiles for other applications	40
Automating tasks with AccessProfiles	41
Creating AccessProfiles that perform automation tasks.	42
Clicking controls	43
Disabling controls	43
Enabling controls	44
Hiding controls	44
Entering text	45
Pausing actions	45
Pressing keys	46
Running plug-ins	46
Closing screens	47
Selecting menu items	47
Generating custom audit logs	48
Editing standard AccessProfiles.	48
Adding tasks	49

Deleting tasks	49
Editing tasks	49
Setting advanced options.	50

Chapter 4. Advanced AccessProfiles	51
---	-----------

Creating advanced AccessProfiles	51
Editing advanced AccessProfiles	53
Editing advanced AccessProfiles from the States diagram	53
Editing advanced AccessProfiles from the General Properties tab.	54
Editing advanced AccessProfiles from the XML Editor tab	54
Creating sample advanced AccessProfiles	54
Login scenario	54
Capturing the signature	56
Creating an application object	56
Creating the state engine	57
Making the state engine work	61
Validating the logon workflow	62
Change password scenario	63
Checking and prompting for relogin	69
Adding fields for random passwords.	70
Using advanced AccessProfiles to meet custom requirements	71
Customized triggers and actions using VBScript and JScript plug-ins	71
Configuring support for existing applications using HLLAPI	73
Signatures	75
About signatures	75
Supported axes	76
Supported types.	76
Available operators.	77
Significance of the root node ('/')	77
Executable signatures	78
Signatures for windows	79
Signatures for web pages.	80
Signatures for HTML	82
Signatures for frames	83
Signatures for Java windows	85
Validating functions	87
Using plug-in API	87
Using plug-in API for customized triggers	87
Using plug-in API for customized actions	87
Plug-in API specifications.	88

Chapter 5. Troubleshooting AccessProfiles	111
Playing back Observer log file	111

Chapter 6. AccessProfiles testing	113
--	------------

Testing AccessProfiles	113
Viewing log descriptions using a sample AccessProfile	114

Chapter 7. Managing AccessProfiles 117

Downloading, uploading, and saving information for AccessStudio 117
 Downloading information 117
 Uploading information 118
 Saving information 118

Chapter 8. Managing authentication services. 119

Associating authentication services with AccessProfiles 119
Creating authentication services 121
Modifying authentication services 124
Managing authentication service groups and group links 124
 Creating authentication service groups and authentication service group links 125
 Modifying authentication service group 126
 Modifying authentication service group links 126

Chapter 9. Managing application objects 127

Creating an application object 127
Modifying an application object 128

Chapter 10. Account data items and templates 131

Account data item templates 131
Account data templates 131
Viewing account data item templates or account data templates 133

Chapter 11. Backing up IMS Server data 135

Chapter 12. Triggers and actions . . . 137

Triggers 137

Actions 152

Chapter 13. Frequently asked questions 163

Notices 165

Glossary 169

A 169
B 170
C 170
D 171
E 172
F 172
G 172
H 172
I 173
J 173
K 173
L 173
M 173
N 174
O 174
P 174
R 175
S 175
T 177
U 177
V 177
W 178

Index 179

Figures

1. AccessStudio workflow	3	6. Sample workflow to enable automatic change password	64
2. Relationship of entities associated with an AccessProfile	5	7. Sample state diagram of change password state engine	69
3. AccessStudio user interface.	8	8. Web signature hierarchy	81
4. Sample workflow to enable automatic logon	55	9. Sample of frameset navigation	82
5. Sample state diagram of Patient Information Manager application.	63	10. Indirect auth-info reference process	120

About this publication

IBM Security Access Manager for Enterprise Single Sign-On AccessStudio Guide provides information about creating and using AccessProfiles. This guide provides procedures for creating and editing standard and advanced AccessProfiles for different application types. It also covers information about managing authentication services and application objects, and information about other functions and features of AccessStudio.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see "Accessibility features" in the *IBM Security Access Manager for Enterprise Single Sign-On Planning and Deployment Guide*.

Technical training

For technical training information, see the following IBM Education website at <http://www.ibm.com/software/tivoli/education>.

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM® Software Support site at <http://www.ibm.com/software/support/probsub.html>.

IBM Security Access Manager for Enterprise Single Sign-On Troubleshooting and Support Guide provides details about:

- What information to collect before contacting IBM Support.
- The various methods for contacting IBM Support.
- How to use IBM Support Assistant.
- Instructions and problem-determination resources to isolate and fix the problem yourself.

Note: The **Community and Support** tab on the product information center can provide additional support resources.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems,

products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Chapter 1. About AccessStudio

AccessStudio is an IBM Security Access Manager for Enterprise Single Sign-On component, which Administrators use to create and manage AccessProfiles.

AccessAgent reads these AccessProfiles and automates the workflow, such as sign-on and sign-off.

The complete solution provides:

- Automatic application account provisioning
- Centralized view of all application accounts
- Automatic sign-on and sign-off
- Authentication management
- User-centric audit logs and report generation
- Centralized de-provisioning for all accounts

See the following topics for more information about AccessStudio:

- “AccessStudio features and benefits”
- “How AccessStudio works” on page 2
- “Creating AccessProfiles in AccessStudio” on page 15
- “AccessStudio basic concepts” on page 3
- “Advanced concepts” on page 5
- “AccessStudio interface” on page 8

AccessStudio features and benefits

AccessStudio provides maximum control over configuring AccessProfiles and managing Authentication Services and their associated data. The associated data includes application objects, authentication services, authentication service groups, and authentication service group links.

You can set up AccessProfiles for the following types of applications:

- Windows applications

Note: Single sign-on support for Microsoft Windows Vista has the following limitations:

- Only auto-injection is supported for the Microsoft Windows Vista and Windows 7 Credential User Interface.
- Auto-capture is not supported for the Microsoft Windows Vista and Windows 7 Credential User Interface.
- The **Automatic Logon** option is not enabled for the Microsoft Windows Vista Credential User Interface. It has the same effect as **Always**.

Important: AccessStudio does not support Windows 120 DPI setting.

- Web applications
- Applications that use Java™ applets
- Terminal applications
- Mainframe applications

- Applications with owner-drawn window screens

AccessStudio offers the following features:

- Standard and advanced modes of AccessProfiles that support requirements of varying complexity.
- Multiple editing with GUI-based and XML editors.
- Flexibility in editing AccessProfiles stored in any location, including AccessProfiles that exists in the IMS Server.
- The ability to import existing AccessProfiles from a local installation of AccessAgent or from the IMS Server.
- Automatic validation of user-configured AccessProfile data to minimize errors.
- The ability to test and debug AccessProfiles.

How AccessStudio works

You can create AccessProfile data and save it to a file in AccessStudio. You can also download and modify AccessProfiles and their associated data from either the IMS Server or the local installation of AccessAgent.

After you create or modify an AccessProfile and its associated data, use the **Upload to IMS** option to publish the data to the IMS Server. After the IMS Server receives the update, the data is downloaded by the AccessAgent associated with the IMS Server. Any changes or new AccessProfiles are applied to the applications in the user system.

The following figure illustrates this process:

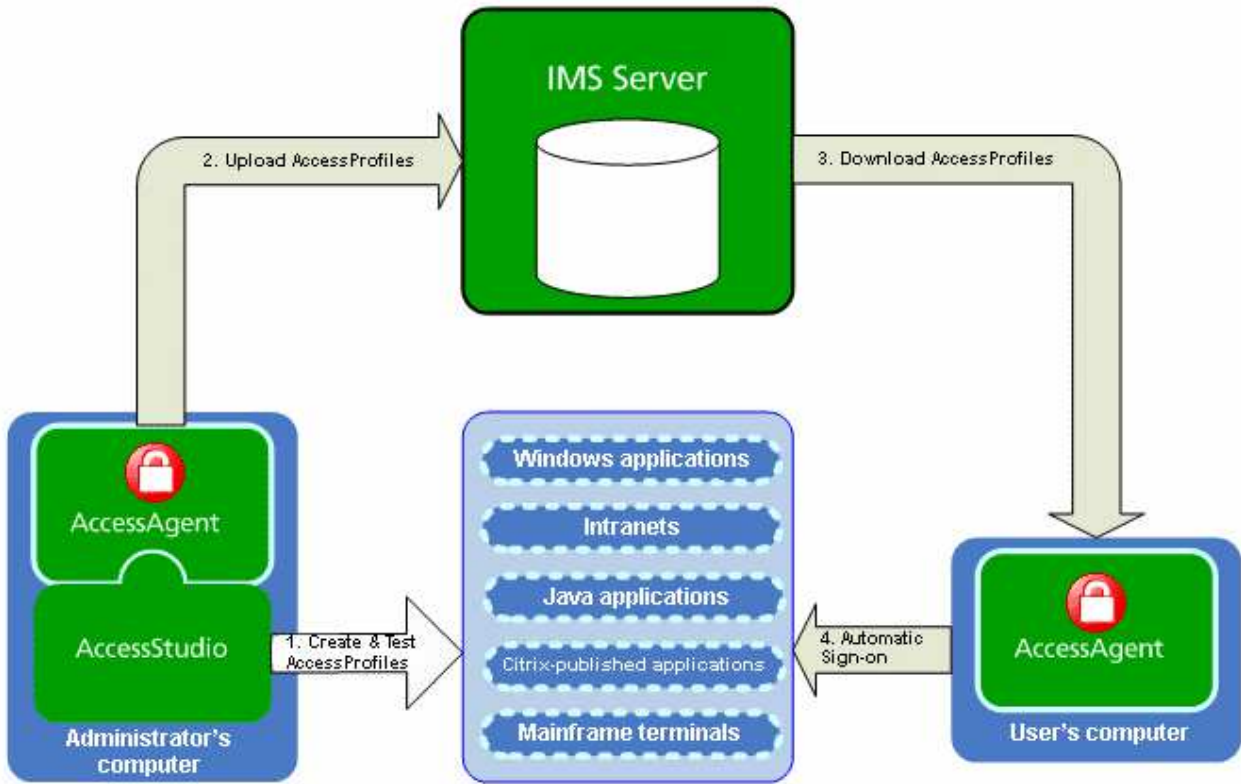


Figure 1. AccessStudio workflow

AccessStudio basic concepts

There are three basic concepts you must understand before you use AccessStudio.

The basic concepts are:

- “AccessProfile”
- “Authentication service” on page 4
- “Application” on page 4

AccessProfile

An AccessProfile contains instructions on handling automation for an application. An application can be an executable file (.EXE) or a web page. An AccessProfile includes:

- Information to identify the application.
- Instructions for automatic operations, such as automatic logon or logoff for the application.
- *Authentication service*, which is the reference to the entity that validates the logon information for the application.
- *Application Object*, which is the reference to the entity that represents the group that is associated with AccessProfile.

There are two kinds of AccessProfiles:

- **Standard AccessProfiles**
Use AccessStudio AccessProfile Generator to create standard AccessProfiles through a series of wizard windows. Use standard AccessProfiles for automating most applications.
- **Advanced AccessProfiles**
For more complex applications, create advanced AccessProfiles. To understand concepts that are used in advanced AccessProfiles, see “Advanced concepts” on page 5. Since Advanced AccessProfiles consists of custom code, custom actions and triggers, IBM does not support Advanced AccessProfiles.

Authentication service

Most applications validate logon information by using a verification entity that is known as an authentication service. All AccessProfiles are associated with an authentication service.

For example, you associate multiple AccessProfiles with a single authentication service. If you associate a group of applications with the same authentication service, AccessStudio applies any change that is made on the logon information in one application, to all applications that are associated with the same authentication service.

The Messaging Software, email Software, and Chat Software are different applications that are represented by different AccessProfiles. However, the same user name and password are used to access all three applications. The Company authentication service validates all logon information. Only one authentication service is created to represent all applications that are validated by the company authentication service.

When you log on to Messaging Software, you do not have to log on again when you access Chat Software or email Software. Your logon information for Messaging Software is captured for all other Company applications, since they are all associated with the same authentication service.

The same concept applies for any changes that are made to your logon information. For example, if you change your password by using email Software, the new password is captured for all other Company applications.

Application

An application object in AccessStudio is a logical representation of a set of executable files (.EXE) or Web pages. An application object can apply tighter control policies for a group of AccessProfiles.

In AccessStudio, you create one AccessProfile for an .EXE file or web page, and the software processes each .EXE file or web page as an application.

An application object handles the grouping of .EXE files and web pages as belonging to the same entity. Associate each AccessProfile with an application object.

The Company authentication service is used by websites mail.example.com, chat.example.com, and by Messaging Software version 5 and version 6. Each .EXE file or web page requires its own AccessProfile. You can create up to four

application objects, depending on the preferred extent of control over the automatic logon policy of the four AccessProfiles.

To have different automatic sign-on mechanisms for mail.example.com and chat.example.com from the two Messaging Software, you can group them under two different application objects. If you also require the same automatic-sign on mechanism for all four, you can group them all under one application object.

Note: You can associate an AccessProfile with only one application object, while one application object can be associated with several AccessProfiles.

The following diagram illustrates the relationship between all entities that are associated with an AccessProfile.

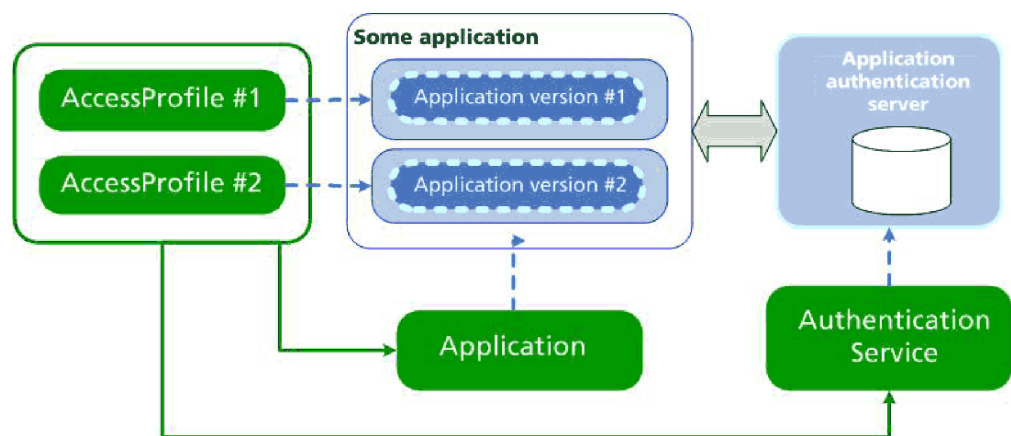


Figure 2. Relationship of entities associated with an AccessProfile

AccessProfiles #1 and #2 represent two different versions of the same application. These applications communicate with an application authentication server. Each AccessProfile representing each application in turn is associated with an application object and an authentication service. The authentication service has a reference to the actual authentication service of the application.

Advanced concepts

You must understand the advanced AccessStudio concepts before you can work on advanced AccessProfiles.

Note: If your application supports the use of the AccessProfile Generator to create AccessProfiles, you can skip this section. Enterprise applications do not support the AccessProfile Generator.

The advanced concepts are:

- “Standard AccessProfile” on page 6
- “Advanced AccessProfile” on page 6
- “State” on page 6
- “Trigger” on page 6
- “Action” on page 6
- “Account data” on page 7

Standard AccessProfile

Standard AccessProfiles, also known as Simple SSO Support, contain all logon, password, and logoff information within one or more screens. Examples are logon screens for applications such as Messaging Software and email Software. Standard AccessProfile also support most applications in different deployment scenarios.

Important: All logon and password change information for each application must be under one AccessProfile.

Advanced AccessProfile

Advanced AccessProfiles, also known as State Engine SSO Support, automates operations that are based on various conditions. Use advanced AccessProfiles for complex logon situations. For example, for verification of conditions before automatic logon, greater control over what triggers an action, and the sequence of these actions.

Advanced AccessProfiles are based on a state engine, which includes states, triggers, and actions. The interaction between these components determines how automatic operations are managed for an AccessProfile. IBM does not support customized Advanced AccessProfiles.

For more information about these three components, see “State,” “Trigger,” and “Action.”

State

A *State* indicates the current condition or status of an application (for example, signed-on status or signed-off status). You can define multiple states and associate triggers that cause a transition from one state to another. It is also possible to provide triggers which point to the same state.

Each state is identified by a user-defined unique ID. You must define a start state to run the transitions of the state.

Trigger

A *trigger* is an event that causes transitions between states in a state engine. For example, a window becomes active or a window is found can be an event which triggers an action. AccessStudio contains predefined triggers.

When a trigger fires, it runs a set of actions that are defined by the Administrator, and then causes transition to the next indicated state. See “Triggers” on page 137 for a list of predefined AccessStudio triggers.

Action

An *action* is the process that is performed in response to a trigger. For example, when the software automatically inserts the user name and password details when the logon window is displayed. When a trigger fires, the actions that are specified for that particular trigger, run in a predefined sequence.

AccessStudio contains predefined actions that can be used to do a set of operations in the application.

The following example describes the interaction between states, triggers, and actions:

1. The *Messaging Software* launches in the Start state.
2. The opening of the logon window fires a trigger followed by the action that automatically inserts the logon information.
3. The messenger comes to the state that is defined in the engine (after the auto fill state).
4. When the user clicks **Sign in**, a trigger is activated. The action to capture the user name and password information occurs.
5. The messenger moves to the after-capture-state.
6. A trigger is activated when the logon window displays the contacts list, and an action to save this user name and password information occurs.
7. The messenger returns to the Start state.

See “Actions” on page 152 for a list of predefined AccessStudio actions.

Account data

Account data is the logon information that is required for verification against an authentication service. The account data is typically the user name, password, and the authentication service that stores the logon information.

AccessStudio stores the account data in a specific format that is known as Account Data Templates. Account data templates provide information about the captured data (for example, which fields are key fields, case-sensitive, and which fields must be hidden).

AccessStudio defines a set of account data template IDs. Each ID represents a particular type of account data.

A set of account data template IDs is defined in AccessStudio with each ID representing a particular type of account data. Account data templates are predefined and you cannot create an account data template. For example, the most commonly used ID (`adt_ciuser_cspwd`) can be specified for applications that have one not case-sensitive user name and one case-sensitive password. For more information, see Chapter 10, “Account data items and templates,” on page 131.

For Company applications, the account data contains the:

- Authentication service ID (which is a user-specified name for the Company authentication service)
- User name
- Encrypted password
- Account data template ID

The account data template ID declares that the user name field is a key field, and that it is not case-sensitive and is not a secret. For the **Password** field, the account data template specifies that it is not a key field, that it is case-sensitive and it is a secret (and requires encryption).

A **key** field is a portion of a record that is used with other key fields to locate a data record in a key file.

AccessStudio interface

The AccessStudio user interface has four main parts.

AccessStudio consists of the following parts:

- “Menu bar”
- “Data type pane” on page 9
- “Details pane” on page 10
- “Message pane” on page 13

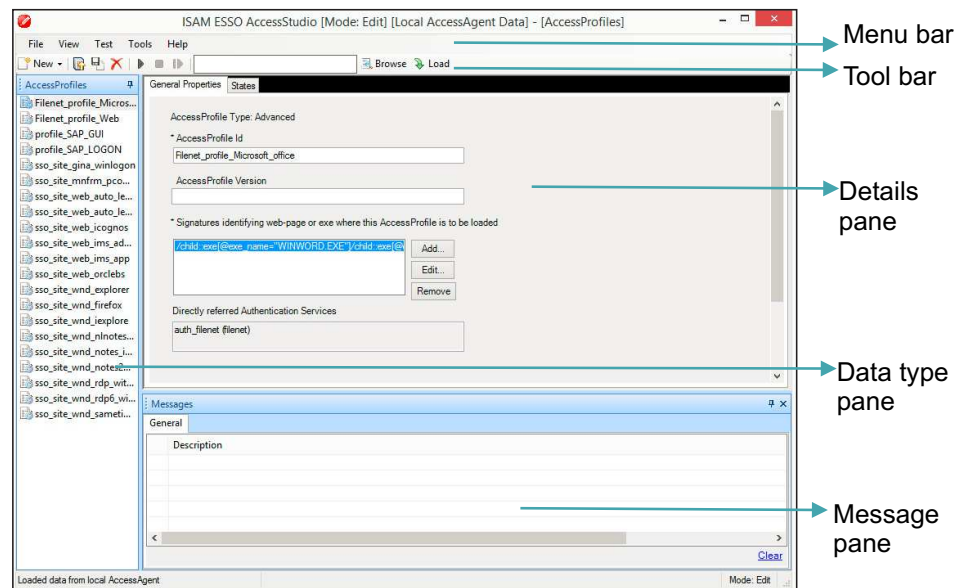


Figure 3. AccessStudio user interface















Menu bar

The AccessStudio menu bar contains the following menus:

Menu	Provides functions for
File	Creating, opening, and saving files that store AccessProfiles. Importing data from the IMS Server or the AccessAgent installed on your computer.
View	Selecting an item (for example, authentication service) for configuration or viewing, or a task for execution (for example, viewing the list of account data templates).
Test	Starting or stopping a test for detecting errors in AccessProfiles and their associated data.
Tools	Saving data that is related to your computer from the IMS Server to a file. Changing your state machine editor font size. Changing your interface language.
Help	Version information of AccessStudio.

Toolbar

The AccessStudio toolbar contains the following menus:

Menu	Descriptions
	Contains functions for creating standard and advanced AccessProfiles, authentication service, application, authentication group, and authentication link.
	Uploads the selected data to the IMS Server.
	Saves the selected data to a file.
	Deletes the selected data. You can also select multiple AccessProfiles, Authentication Services, Applications, Advanced Date and click the Delete icon.
	Starts test.
	Stops test.
	Restarts test.
	Locates an Observer log file.
	Displays the contents of the Observer log file in the AccessStudio Real-Time Logs pane.
	Starts the Observer log file playback.
	Pauses the playback.
	Resumes the paused playback.
	Pauses the playback and manually go to the next log entry.
	Stops the playback.
Playback speed <input type="text" value="Normal"/> ▾	Reduces the speed of the playback.

Data type pane

The Data type pane displays AccessProfiles and their associated data. From the Data type pane, you can view all authentication services, application objects, authentication service groups, authentication service group links, account data templates, and account data item templates.

For example, when you access an AccessProfile through the **View** menu, the Data type pane displays a list of data items.

Right-click on a list item to complete either of the following actions: upload AccessProfiles to the IMS Server, save to a file, or delete an AccessProfile.

Note: The Data type pane displays AccessStudio data, such as AccessProfiles, Authentication Services, Application, Authentication Service Groups, Authentication Service Group Links, Account Data Templates, and Account Data Item Templates. Use the Details pane to edit existing list item information.

Details pane

You can create AccessProfiles and configure these profiles to suit your workflow requirements. The Details pane contains tabs that display different options for AccessProfiles, Authentication Services, Applications, and Account Data items and templates

General Properties tab

Use the **General Properties** tab to view and edit the summary information about the AccessProfile that is selected from the Data type pane. You can also specify extra support information and advanced options in the **General Properties** tab.

AccessProfile summary information

Use the **General Properties** tab to edit the AccessProfile id and version. You can also add, edit, and delete signatures and tasks.

Extra Support Information

Use the **Extra Support Information** fields to enable screen scraping support for terminal and mainframe applications, HLLAPI support for mainframe applications, and Java™ applications.

Advanced Options

Use the **Advanced Options** to edit the following fields:

The value that you entered contains invalid character.

Options	Description
Signatures	Manage the signatures that identify the web page or Windows application where an AccessProfile is loaded. You can add, edit, and delete a signature.
Application Id	Select the application object that you want to associate with the AccessProfile. See "Creating an application object" on page 127.
Only run when SSO is enabled	Specify whether to load the AccessProfile into an application only when single sign-on is enabled.
Behaviour on AccessAgent log-on	Specify whether you want the AccessProfile to do the following on AccessAgent log-on: <ul style="list-style-type: none">• Reset state machine• Reset local properties• Reset global properties

Options	Description
AccessProfile compatibility information	<p>Minimum AA version Enter the minimum version of AccessAgent that is compatible with the AccessProfile.</p> <p>Order of variance Enter the variance between the minimum AccessAgent version and the AccessAgent version used.</p> <p>Supported Platforms Select the platform where you want to deploy the AccessProfile. To select multiple platforms, press Ctrl and click the name of the platform.</p>

Description

Use the **Description** field to enter the details of the AccessProfile.

States tab

Use the **States** tab to organize states, triggers, and actions to build a state-machine. The **States** tab also displays the **Properties** pane that contains the following AccessProfile properties: **Form Editor**, **XML editor**, and **XML viewer**.

States diagram toolbar

Use the states diagram toolbar to add, cut, copy, paste, or delete a state, trigger, or action.

Options	Description
New state	Click the New state icon to add a state.
New trigger	Click the New trigger icon to add a trigger.
New action	Click the New action icon to add an action.
Add Widget	Click the Add Widget icon to add an AccessProfile widget.
Select	Click the Select icon to view the states and the corresponding triggers and actions in it.
Select by type	Click the Select by type icon to view by trigger, by action type, or by bag id.
Fade all states	Click the Fade all states icon to change the color of all states to gray. Click a state to highlight the item and to edit its properties on the Properties pane.
Darken all states	Click the Darken all states icon to display all the states in default color.
Collapse	Select the state, trigger, or action that you are not interested in, then click the Collapse icon to hide it. You can either collapse all or collapse only the faded states. This option helps you focus on the state, action, or trigger you are working on, and reduces the clutter in the state-machine diagram. You can also point to a collapsed trigger or action to see a tooltip summary of its contents, instead of looking at the details in the Form Editor .

Options	Description
Expand	Select the state, trigger, or action you must work on and click the Expand icon to show the collapsed or the darkened items. You can either expand all or expand only the faded items. With this option, you can focus on the selected state, trigger, or action and also improve the display in the state-machine diagram. You can also point over an expanded trigger or action to see a tooltip summary of its contents, instead of viewing the details in the Form Editor .
Cut	Click the Cut icon to cut a state, trigger, or action.
Copy	Click the Copy icon to copy a state, trigger, or action.
Paste	Click the Paste icon to paste a state, trigger, or action.
Delete	Click the Delete icon to delete a state, trigger, or action.
Undo	Click the Undo icon to erase the recent changes.
Replace Text	Click the Replace Text icon to replace the underlying XML of the AccessProfile, such as signature fragments, or property names. Note: Do not search or replace keywords like 'trigger', 'wnd_click_action' or 'state' because it might break the XML structure.
Save state-machine as an image	Click the Save state-machine as an image icon to save the state-machine as an image.

Properties pane

Use the **Properties** pane to edit the states, triggers, and actions.

- **Form Editor tab**

Use the **Form Editor** tab to add, edit, and view configuration information that is related to the list item selected from the Data type pane.

Note: The **Form Editor** tab contents in the **Properties** pane changes when you select a state, trigger, or action. Specify the information in the **General Properties** tab before you edit the fields in the **Form Editor** tab.

Important: In the AccessProfiles offered by IBM, the descriptions about the states, actions, and triggers are intended for Advanced AccessProfile users. These descriptions improve the usage and maintenance of the AccessProfile.

Regardless of the locale set on your machine, the descriptions in the user interface are displayed only in the English language. However, you can edit or modify the descriptions in the AccessProfile. Changes to the descriptions do not affect the behavior or performance of the AccessProfile.

- **Conditions**

Click the **Conditions** twistie to expand the category. Use the **Add Condition** field to add either **Check a script return value condition** or **Check a property value condition**.

- **Advanced Options**

Expand the **Advanced Options** twistie and use the fields to indicate whether to queue actions or use for sign-in.

- **Capture fields**

Expand the **Capture fields** twistie to add and capture signatures for Windows, Web, Windows combo box, Windows List-box, Windows List-view, and Java application controls.

Note: The **Windows control (Keyboard input)** and **Currently focused control** are deprecated.

- **Injection fields**

Expand the **Injection fields** twistie to inject credentials for Windows, Web, Windows combo box, Windows List-box, Windows List-view, and Java application controls.

- **Auth Info**

Expand the **Auth Info** twistie and use the fields to add indirect authentication information for Windows signatures, Web, keyboard input, Windows combo-box, Windows list-box, Mainframe screen output, Java window signature, and direct authentication group information.

- **Random password fields**

Expand the **Random password fields** twistie to generate random password items during password change scenarios for Windows, Web, Windows combo box, Windows List-box, Windows List-view, and Java application controls.

- **Add search filter**

Expand the **Add search filter** twistie to add search filter items for Windows, Web, Windows combo box, Windows List-box, Windows List-view, and Java application controls.

- **Advanced Options**

Expand the **Advanced Options** twistie and use the fields to retrieve account data from the Wallet, empty account data bag, override injection policy, and use local bag. You can also capture the signature of the window under which injection happens.

- **XML Editor tab**

Use the **XML Editor** tab to modify the actual XML code of the AccessProfile or its associated data. Triggers or actions with erroneous XML structures are marked with a red exclamation point on the **Data type** pane.

Important: The **XML Editor** is used for advanced functions. Use this tab only if you are an AccessStudio expert and familiar with XML. Do not copy and paste data in the **XML** tab because each ID of a state, trigger, or action must be unique. Copying and pasting replicate IDs that cause schema and state-machine errors.

Note: AccessStudio experts can also click the **Replace Text** icon in the **States** tab toolbar to find and replace XML data in the XML Editor.

- **XML Viewer tab**

Use the **XML Viewer** tab to view the actual XML code of a list item.

Message pane

Use the Messages pane to view real-time information about the currently active task. To view the **Messages** panes, click the **View menu > Messages**. The Message pane also hosts the variable values for every parameter that is associated with each line of the log file.

Note: Each pane has a **Push-pin** icon. Click the **Push-pin** icon on the toolbar of the pane you want to hide. When hidden, the pane name displays as a small tab. Click this tab to show the pane. You can also put the cursor over the collapsed edge to open up the hidden item until the cursor moves out of the window area.

Chapter 2. AccessProfiles

Each application is represented by an *AccessProfile*, which is a set of instructions that define the automatic logon mechanism for that particular application.

Administrators create AccessProfiles for third-party applications to provide single sign-on support for more applications.

Creating AccessProfiles in AccessStudio

Create and maintain AccessProfiles for your applications with AccessStudio.

The major steps in creating or maintaining AccessProfiles in AccessStudio are as follows:

Table 1. Procedure for creating or maintaining AccessProfiles

What to do	Where to find information
Create an AccessProfile.	<ul style="list-style-type: none">• Chapter 3, “Standard AccessProfiles,” on page 17• Chapter 4, “Advanced AccessProfiles,” on page 51
Customize an AccessProfile.	<ul style="list-style-type: none">• “Automating tasks with AccessProfiles” on page 41• “Using advanced AccessProfiles to meet custom requirements” on page 71
Test the AccessProfile.	Chapter 6, “AccessProfiles testing,” on page 113
Upload the AccessProfile to the IMS Server.	“Uploading information” on page 118

Chapter 3. Standard AccessProfiles

Use AccessStudio to create, import, and view the existing AccessProfiles from the IMS Server or AccessAgent installed on your computer.

Note: Before you use AccessStudio, review the concepts that are provided in Chapter 1, “About AccessStudio,” on page 1.

Standard AccessProfiles, also known as Simple SSO Support, contains all logon, password, and logoff information within single or multiple screens. Examples are the logon screens for applications, such as IBM Sametime® and *CompanyMail*. Standard AccessProfiles also support most applications in different deployment scenarios.

You can create standard AccessProfiles through the AccessProfile Generator.

- “Creating standard AccessProfiles for different application types”
- “Automating tasks with AccessProfiles” on page 41
- “Editing standard AccessProfiles” on page 48

Creating standard AccessProfiles for different application types

When you create an AccessProfile using the AccessProfile Generator, the wizard automatically creates the application object and the authentication service for the AccessProfile. Use the AccessProfile Generator to create different types of AccessProfiles:

For more information on application objects and authentication services, see “AccessStudio basic concepts” on page 3.

- “Creating AccessProfiles for Windows applications”
- “Creating AccessProfiles for Web applications” on page 23
- “Creating logon AccessProfiles for Java applications” on page 28
- “Creating logon AccessProfiles for applications or applets that use Java” on page 30
- “Creating AccessProfiles for terminal applications” on page 31
- “Creating AccessProfiles for Mainframe or cursor-based applications” on page 33
- “Creating logon AccessProfiles for Mainframe applications with HLLAPI support” on page 38
- “Creating logon AccessProfiles for other applications” on page 40

Creating AccessProfiles for Windows applications

Windows applications (for example, Win32, 16-bit) such as *Company Mail* are applications that run on the Windows platform.

About this task

Creating AccessProfiles for Windows applications requires the following tasks:

- “Creating a logon AccessProfile for Windows applications” on page 18
- “Creating a change password AccessProfile for Windows applications” on page 20

- “Creating a logoff AccessProfile for Windows applications” on page 21
- “Creating an “other task” AccessProfile for Windows applications” on page 22

Creating a logon AccessProfile for Windows applications

Use the AccessProfile Generator to create a logon AccessProfile for an application that runs on Windows.

About this task

Captured logon fields are translated into signatures. Signatures contain XML Path Language, a language that facilitates XML document navigation to select elements and attributes.

IBM Security Access Manager for Enterprise Single Sign-On uses signatures to identify application screens and Windows elements. These signatures are then communicated into AccessAgent.

The next time that the same fields are presented, AccessAgent automatically supplies the user credentials in their respective fields.

For more information about Signatures, see “Creating advanced AccessProfiles” on page 51.

Procedure

1. Open AccessStudio by selecting **Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**.
2. Start the AccessProfile Generator by clicking the **New** icon.
3. Select **New AccessProfile (using Generator)**.
4. Click **Next** from the Welcome window.
5. Launch the application for which you want to create an AccessProfile.
6. After the application screen opens, click **Next** to proceed.
7. Enter a unique name for the application in the **Application name** field.
8. Select **Windows application** as the application type.
9. Click **Next**.
10. Select the **Logon** radio button.
11. Click **Next**.
12. Enter a unique name in the **Enter a name** field for the screen you want to capture.
13. Based on your selected task, capture identification information for the fields on the application screen.

Note: The fields available for each automated task vary.

- a. Click the **Finder tool** from the AccessProfile Generator.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. Drag the **Finder tool** on the matching field in the application screen.

As you drag the **Finder tool** to the application, the AccessProfile Generator selects the field or button you want to capture.

- c. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear** option is activated. The default screen name from the application is retrieved. Click **Clear** to undo the capture.
 - d. Use the **Extra Field Finder tool** to capture an additional field unique to the application, if available for the selected task.
The **Extra Field** can be a list or any domain field group.
 - e. Click **Edit Signature** and click **Highlight Control** to confirm the captured signature.
 - f. Click **Advanced Settings** to perform the task only when a certain condition is satisfied.
For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Editing standard AccessProfiles” on page 48.
14. After capturing all the application fields, click **Next** to proceed.
 15. Optional: Select the logon screen from the **Screens identified** field and choose from the following optional steps:
 - To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.
 - For logon screens, select the default settings for similar application screens. The options are: **Ask User**, **Do not auto-fill**, **Auto-fill**, and **Auto-fill and submit**.
Based on the selection, AccessStudio automates or does not automate similar logon screens as set in this AccessProfile.
 16. Click **Next**.
 17. Specify whether you want AccessStudio to identify the successful logon. Choose from the following options:
 - **No**. If you select this option, no success screen or message displays.
 - **Yes, identify the screen that appears upon successful logon**. If you select this option, drag the **Finder tool** and drop it on the success application screen.
When the **Finder tool** is positioned over the screen, release the mouse button.
Based on the captured item, you can also modify the screen title or text.
 - **Yes, simply detect closure of the logon screen**.
If you select this option, the logon screen closes without any confirmation.
 18. Click **Next**.
 19. Do one of the following options:
 - Select **Use a previously created authentication service** and choose an authentication service from the list.
 - Select the default **Create one for me automatically** option to create an authentication service.
 20. Click **Finish** to return to the AccessStudio user interface.
The captured tasks and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

21. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation is displayed.

Another message box is displayed, indicating the success or failure of the upload.

Creating a change password AccessProfile for Windows applications

Use the AccessProfile Generator to create a change password AccessProfile for an application that runs on Windows.

Procedure

1. Follow the steps from Open AccessStudio to Select Windows application of the “Creating a logon AccessProfile for Windows applications” on page 18 procedure.
2. If the Standard access profile exists already, follow the steps that are specified in the “Setting advanced options” on page 50.
3. Select **Change password**.
4. Click **Next**.
5. Enter a unique name for the screen you want to capture.
6. Capture identification information for the fields on the application screen.
 - a. Click the **Finder tool** from the AccessProfile Generator.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. Drag the **Finder tool** to the corresponding field in the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
 - c. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear** option is activated. The default screen name from the application is retrieved. Click **Clear** to undo the capture.
 - d. Click **Advanced Settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Editing standard AccessProfiles” on page 48.
7. Click **Next**.
8. Select the **Change Password screen** from the **Screens identified** field.
 - a. To edit the previously captured screen, select the screen and click **Next**.
 - b. To remove the previously captured screen, select the screen title in the list box and click **Delete**.
9. Specify whether you want AccessStudio to identify the successful changing of the password.
 - The options are **No** (no success screen or message displays), **Yes, identify the screen that appears upon successful change password**, and **Yes, simply detect closure of the change password screen**.
 - If you selected **Yes, identify the screen that appears upon successful change password**, drag the **Finder tool** and drop it on the success application screen.

When the **Finder tool** is positioned over the screen, release the mouse button.

Based on the captured item, you can also modify the screen title or text.

10. Click **Finish** to return to the AccessStudio user interface.

The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

11. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation is displayed.

Another message box is displayed, indicating the success or failure of the upload.

Creating a logoff AccessProfile for Windows applications

Use the AccessProfile Generator to create a logoff AccessProfile for an application that runs on Windows.

Procedure

1. Follow the steps from Open AccessStudio to Select Windows application of the “Creating a logon AccessProfile for Windows applications” on page 18 procedure.
2. Select **Logoff**.
3. Click **Next**.
4. Select a logoff method. You can either select **Close the application** or **Log off gracefully**.
 - If you choose to close the application, proceed to Upload the AccessProfile to the IMS Server step.
 - If you choose to log off gracefully, select a graceful logoff option and proceed to the next step.
5. Specify whether there is a specific set of screens or only generic screens for the graceful logoff task.
 - To specify a specific set of screens and actions, select **Identify specific screens and set specific actions**, click **Next**, and go to the next step.
 - To specify screens that do not need a custom set of actions, select **Create a generic set of actions for unidentified screens**, click **Next**, and go to Specify actions for logoff step.
6. If you chose **Identify specific screens and set specific actions**, specify the following details:
 - a. Identify the logoff screen by entering a unique name for the screen capture.
 - b. Click the **Finder tool** from the **Unique screen text for identification** field.

Note: If the **Finder tool** is not enabled, save the AccessProfile as it is, then restart AccessStudio.

- c. Drag the **Finder tool** on the matching field in the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.

- d. When the **Finder tool** is positioned over the field, release the mouse button.
 - e. Click **Advanced settings** to perform the task only when a certain condition is satisfied. For more information, see “Creating AccessProfiles that perform automation tasks” on page 42.
 - f. Click **Next**.
7. Specify actions for logoff.

Select the action that you want to automate from the **Available actions** list. See “Creating AccessProfiles that perform automation tasks” on page 42 for details.

 - a. Select each action, enter a menu path, or use the **Finder tool**.
 - b. Click **Add**.
 - c. Click **Next** after adding all the required logoff actions.
 8. Identify the logoff screen.

Select the logoff screen that you have captured from the **Screens (last screen if checked)** field.

 - To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.
 9. Click **Finish** to return to the AccessStudio user interface.

The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.
 10. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation is displayed.

Another message box is displayed, indicating the success or failure of the upload.

Creating an "other task" AccessProfile for Windows applications

Use the AccessProfile Generator to create a uniqueAccessProfile for an application that runs on Windows.

Procedure

1. Follow the steps from Open AccessStudio to Select Windows application of the “Creating a logon AccessProfile for Windows applications” on page 18 procedure.
2. Select **Other tasks** as the task to automate.
3. Click **Next**.
4. Enter a unique name for the other task screen to capture.
5. Based on your selected task, capture identification information for the fields on the application screen.
 - a. Click the **Finder tool**, drag to the corresponding fields on the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.

- b. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear signature** option is activated.
 - c. The default screen name from the application is retrieved. Click **Clear signature** to undo the capture.
 - d. Click **Advanced settings** to perform the task only when a certain condition is satisfied. For more information, see “Creating AccessProfiles that perform automation tasks” on page 42.
6. Click **Next**.
 7. Specify actions for the task.
 - a. Select from the list of available actions. See “Creating AccessProfiles that perform automation tasks” on page 42 for details.
 - b. Click **Add**.
 8. Click **Next**.
 9. Identify the task automation screen. Select the screen that you have captured from the **Screens identified** field.
 - To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.
 10. Click **Finish** to return to the AccessStudio user interface.
The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.
 11. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation is displayed.

Another message box is displayed, indicating the success or failure of the upload.

Creating AccessProfiles for Web applications

Use the AccessProfile Generator for set profiles for Web applications running in web browsers.

About this task

See the procedures for creating AccessProfiles for Web applications:

- “Creating a logon AccessProfile for Web applications”
- “Creating a change password AccessProfile for Web applications” on page 25
- “Creating a logoff AccessProfile for Web applications” on page 26
- “Creating an “other task” AccessProfile for Web applications” on page 27

Creating a logon AccessProfile for Web applications

Use the AccessProfile Generator to create a logon AccessProfile for an application that runs on a web browser.

About this task

Note: Auto-logon to AccessAdmin is not supported for Mozilla Firefox web browser.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Click **Next** from the Welcome window.
4. After starting the AccessProfile Generator, open the application logon screen that requires an AccessProfile.
5. After opening the application screen or web page, click **Next**.
6. Enter a unique **Application name**.
7. Select **Web application** as the application type.
8. Click **Next**.
9. Select **Logon** as the task to automate.
10. Click **Next**.
11. Enter a unique name for the screen or web page to capture.
12. Based on your selected task, capture identification information for the fields on the application screen.

Note: The fields available for each automated task varies.

- a. Click the **Finder tool** from the AccessProfile Generator.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. Drag the **Finder tool** to the corresponding field in the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
- c. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear** option is activated. The default screen name from the application is retrieved. Click **Clear** to undo the capture.
- d. Use the **Extra field Finder tool** to capture an additional field unique to the application, if available for the selected task.
- e. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see "Creating AccessProfiles that perform automation tasks" on page 42 or "Setting advanced options" on page 50.

13. After capturing all the application fields, click **Next >**.
14. Select the logon screen from the **Screens identified** field.
 - To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.
 - For logon screens, select the default settings for similar application screens. The options are: **Ask User**, **Do not auto-fill**, **Auto-fill**, and **Auto-fill and submit**.

Based on the selection, AccessStudio automates or does not automate similar logon screens as set in this AccessProfile.

15. Click **Next**.
16. Specify whether you want AccessStudio to identify the successful logon. Choose from the following options:
 - **No.** If you select this option, no success screen or message displays.
 - **Yes, identify the screen that appears upon successful logon.**
If you select this option, drag the **Finder tool** and drop it on the success application screen or web page. When the **Finder tool** is positioned over the screen or web page, release the mouse button.
Based on the captured item, you can also modify the screen title or text.
 - **Yes, simply detect closure of the logon screen.**
If you select this option, the logon screen closes without any confirmation.
17. Click **Next**.
18. Select or create an authentication service.
If there was no previously created authentication service, leave the default selection at **Create one for me automatically**.
19. Click **Finish** to return to the AccessStudio user interface.
The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

20. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating a change password AccessProfile for Web applications

Use the AccessProfile Generator to create a change password AccessProfile for an application that runs on a web browser.

Procedure

1. Follow the steps from Open AccessStudio to Select Web application of the “Creating a logon AccessProfile for Web applications” on page 23 procedure.
2. Select **Change password** as the task to automate.
3. Enter a unique name for the screen or web page to capture.
4. Click **Next**.
5. Based on your selected task, capture identification information for the fields on the application screen.
 - a. Click the **Finder tool** and drag to the corresponding fields on the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
 - b. When the **Finder tool** is positioned over the field, release the mouse button.
 - c. Click **Advanced settings** to perform the task only when a certain condition is satisfied. For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

Note: The fields available for each automated task varies.

6. Click **Next**.
7. Identify successful password change.
 - a. You can either select **Yes, identify the screen that appears upon successful change password** or **Yes, simply detect closure if the change password screen**.
 - b. Click the **Finder tool** from the **Unique screen text for identification** field, then drag the icon on the matching field in the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
 - c. When the **Finder tool** is positioned over the field, release the mouse button.
8. Click **Finish** to return to the AccessStudio user interface.

The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, "AccessProfiles testing," on page 113.

9. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating a logoff AccessProfile for Web applications

Use the AccessProfile Generator to create a logoff AccessProfile for an application that runs on a web browser.

Procedure

1. Follow the steps from Open AccessStudio to Select Web application of the "Creating a logon AccessProfile for Web applications" on page 23 procedure.
2. Select **Logoff** as the task to automate.
3. Click **Next**.
4. Select a logoff method. You can either select **Close the application** or **Log off gracefully**.
 - If you choose to close the application, select **Close the application** and proceed to the Upload the AccessProfile to the IMS Server step.
 - If you choose to log off gracefully, select a graceful logoff option and proceed to the next step.
5. Specify whether there is a specific set of screens or only generic screens for the graceful logoff task.
 - To specify a specific set of screens and actions, select **Identify specific screens and set specific actions**, click **Next**, and go to the next step.
 - To specify screens that do not need a custom set of actions, select **Create a generic set of actions for unidentified screens**, click **Next**, and go to the Specify actions for logoff step.
6. If you selected **Identify specific screens and set specific actions**, specify the following details.
 - a. Identify screen for logoff.

Enter a unique name for the screen you are about to capture.

- b. Click the **Finder tool** from the **Unique screen text for identification** field, then drag the icon on the matching field in the application screen. As you drag the **Finder tool** to the application, the AccessProfile Generator marks the field or button that can be captured.
- c. When the **Finder tool** is positioned over the field, release the mouse button.
- d. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

- e. Click **Next**.
7. Specify actions for logoff.
- Select the action that you want to automate from the **Available actions** list. See “Creating AccessProfiles that perform automation tasks” on page 42.
- a. Select each action, enter a menu path, or use the **Finder tool**.
 - b. Click **Add**.
 - c. Click **Next** after adding all the required logoff actions.

8. Identify the logoff screen.
- Select the logoff screen that you have captured from the **Screens (last screen if checked)** field.
- To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.

9. Click **Finish** to return to the AccessStudio user interface.
- The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

10. Upload the AccessProfile to the IMS Server to activate it.
- a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating an "other task" AccessProfile for Web applications

Use the AccessProfile Generator to create a unique AccessProfile for an application that runs on a web browser.

Procedure

1. Follow the steps from Open AccessStudio to Select Web application of the “Creating a logon AccessProfile for Web applications” on page 23 procedure.
2. Select **Other tasks** as the task to automate.
3. Click **Next**.
4. Enter a unique name for the other task screen to capture.
5. Based on your selected task, capture identification information for the fields on the application screen.
 - a. Click the **Finder tool**, drag to the corresponding fields on the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.

- b. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear signature** option is activated.
- c. The default screen name from the application is retrieved. Click **Clear signature** to undo the capture.
- d. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

6. Click **Next**.
7. Specify actions for the task.
Select from the list of available actions. See “Creating AccessProfiles that perform automation tasks” on page 42 for details.
8. Click **Add**.
9. Click **Next**.
10. Identify the task automation screen.
Select the screen that you captured from the **Screens identified** field.
 - To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.
11. Click **Finish** to return to the AccessStudio user interface.
The captured task and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.
12. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation displays.
Another message box displays, indicating the success or failure of the upload.

Creating logon AccessProfiles for Java applications

Use the AccessProfile Generator for set logon profiles for Java applications.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Click **Next** from the Welcome window.
4. Open the application logon screen that requires an AccessProfile.
5. When you open the application screen or web page, click **Next**.
6. Enter a unique name for the application and select **Java application** as the application type.
7. Click **Next** twice.
8. Enter a unique name for the screen or web page to capture.

9. Based on your selected task, capture identification information for the fields on the application screen.

Note: The fields available for each automated task varies.

- a. Click the **Finder tool** from the AccessProfile Generator, then drag the icon on the matching field in the application screen or web page.
 - b. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
 - c. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear** option is activated. The **screen title** field retrieves its default screen name from the application. Click **Clear** to undo the capture.
 - d. After capturing all the application fields, click **Next**.
10. Optional: Select the logon screen from the **Screens identified** field and choose from the following optional steps:
 - To edit the previously captured screen, select the screen and click **Next**.
 - To remove the previously captured screen, select the screen title in the list box and click **Delete**.
 - For logon screens, select the default settings for similar application screens. The options are: **Ask User**, **Do not auto-fill**, **Auto-fill**, and **Auto-fill and submit**.
Based on the selection, AccessStudio automates or does not automate similar logon screens as set in this AccessProfile.
 11. Click **Next**.
 12. Specify whether you want AccessStudio to identify the successful logon. Choose from the following options:
 - **No**. If you select this option, no success screen or message displays.
 - **Yes, identify the screen that appears upon successful logon**.
If you select this option, drag the Finder tool and drop it on the success application screen or web page. When the **Finder tool** is positioned over the screen or web page, release the mouse button.
Based on the captured item, you can also modify the screen title or text.
 - **Yes, simply detect closure of the logon screen**.
If you select this option, the logon screen closes without any confirmation.
 13. Click **Next**.
 14. Do one of the following options:
 - Select **Use a previously created authentication service** and choose an authentication service from the list.
 - Select the default **Create one for me automatically** option to create an authentication service.
 15. Click **Finish** to return to the AccessStudio user interface.
The captured tasks and the identified screens are displayed in the **General Properties** tab.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 4, “Advanced AccessProfiles,” on page 51 or Chapter 6, “AccessProfiles testing,” on page 113 for details.

16. Upload the AccessProfile to the IMS Server to activate it.

- a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
- b. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating logon AccessProfiles for applications or applets that use Java

Use the AccessProfile Generator for set logon profiles for applications or applets that use Java.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Click **Next** from the Welcome window.
4. Open the application logon screen that requires an AccessProfile.
5. After you open the application screen or web page, click **Next**.
6. Enter a unique name for the application and select **Java applet** as the application type.
7. Click **Next** twice.
8. Enter a unique name for the screen or web page to capture.
9. Based on your selected task, capture identification information for the fields on the application screen.

Note: The fields available for each automated task varies.

- a. Click the **Finder tool** from the AccessProfile Generator, then drag the icon on the matching field in the application screen or web page.
 - b. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
 - c. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear** option is activated. The **screen title** field retrieves its default screen name from the application. Click **Clear** to undo the capture.
 - d. After you capture all the application fields and the web element, click **Next**.
10. Select the actions to use for logging on.
Highlight each option and click **Add**.
 11. Click **Next**.
 12. Select or create an authentication service.
If there was no previously created authentication service, leave the default selection at **Create one for me automatically**.
 13. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload the profiles to the IMS Server. For more information, see Chapter 6, "AccessProfiles testing," on page 113.

14. Upload the AccessProfile to the IMS Server to activate it.

- a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
- b. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating AccessProfiles for terminal applications

Terminal emulator, terminal application (TTY) is a program that emulates a video terminal within a display architecture. Though typically synonymous with a command line shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window.

About this task

See the procedures for creating AccessProfiles for terminal applications:

- “Creating a logon AccessProfile for terminal applications”
- “Creating a change password AccessProfile for terminal applications” on page 32

Example

Examples of TTY applications are PuTTY and SecureCRT.

Creating a logon AccessProfile for terminal applications

Use the AccessProfile Generator to create a logon AccessProfile for an application that runs on a terminal window.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Click **Next** from the Welcome window.
4. Open the application logon screen that requires an AccessProfile.
5. Click **Next**.
6. Enter a unique name for the application and select **Terminal** as the application type.
7. Click **Next**.
8. Select **Logon** as the task to automate.
9. Click **Next**.
10. Enter a unique name for the screen.

Note: The fields available for each automated task varies.

11. Based on your selected task, capture identification information for the fields on the application screen.
 - a. Click the **Finder tool** from the AccessProfile Generator.

Note: If the **Finder tool** is not enabled, click the **Edit signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. Drag the **Finder tool** to the corresponding field in the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.

- c. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear** option is activated. The default screen name from the application is retrieved. Click **Clear** to undo the capture.
 - d. Click **Advanced settings** to perform the task only when a certain condition is satisfied.
For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.
12. After you capture all the application fields, click **Next**.
 13. Select each action to include for logging on in the **Available actions** group and moving it to the **Selected actions** group.
 - a. Select actions from the **Available actions** pane.
 - b. Click the **Add** button.
 - c. Use the **Move Up** and the **Move Down** icons to arrange the sequence of actions in the **Selected actions** pane.
 14. Click **Next**.
 15. Specify whether you want AccessStudio to identify the successful logon to avoid capturing incorrect credentials.
If you selected **Yes**, enter a unique case-sensitive screen text for identification.
 16. Click **Next**.
 17. Select or create an authentication service.
If there was no previously created authentication service, leave the default selection at **Create one for me automatically**.
 18. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

19. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation displays.
Another message box displays, indicating the success or failure of the upload.

Creating a change password AccessProfile for terminal applications

Use the AccessProfile Generator to create a change password AccessProfile for an application that runs on a terminal window.

Procedure

1. Follow the steps from Open AccessStudio to Enter a unique name for the application of the “Creating a logon AccessProfile for terminal applications” on page 31 procedure.
2. Select **Change password** as the task to automate.
3. Click **Next**.
4. Enter a unique name for the screen.
5. Capture the password change information about the application screen.
 - a. Click the **Finder tool**, drag to the corresponding fields on the application screen. As you drag the **Finder tool** to the application, the AccessProfile Generator marks the field or button that can be captured.

- b. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear signature** option is activated.
- c. The default screen name from the application is retrieved. Click **Clear signature** to undo the capture.
- d. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

- 6. Click **Next**.
- 7. Select each action to include for changing password in the **Available actions** group and moving it to the **Selected actions** group.
 - a. Select actions from the **Available actions** pane.
 - b. Click the **Add** button.
 - c. Use the **Move Up** and the **Move Down** icons to arrange the sequence of actions in the **Selected actions** pane.
- 8. Click **Next**.
- 9. Identify the successful password change screen.
 - The options are **No** (no successful change password screen displays), and **Yes** (a successful change password screen displays).
 - If you select **Yes**, enter the text that displays at the command prompt upon successful change password in the **Unique screen text for identification** field.
- 10. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

- 11. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating AccessProfiles for Mainframe or cursor-based applications

Mainframe applications that are typically run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

About this task

See the procedures for creating AccessProfiles for Mainframe or cursor-based applications.

- “Creating a logon AccessProfile for Mainframe or cursor-based applications” on page 34
- “Creating a change password AccessProfile for Mainframe or cursor-based applications” on page 35
- “Creating a logoff AccessProfile for Mainframe or cursor-based applications” on page 36

Creating a logon AccessProfile for Mainframe or cursor-based applications

Use the AccessProfile Generator to create a logon AccessProfile for a Mainframe or cursor-based application.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Open the application logon screen that requires an AccessProfile.
4. Click **Next**.
5. Enter a unique name for the application and select **Mainframe or cursor-based application** as the application type.
6. Click **Next**.
7. Specify the task that you want to automate by selecting the appropriate radio button. For this procedure, select **Logon**.
8. Click **Next**.
9. Enter a unique name for the screen.
10. Based on your selected task, capture identification information for the fields on the application screen.
 - a. Click the **Finder tool** from the AccessProfile Generator.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. Drag the **Finder tool** to the corresponding field in the application screen. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
 - c. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear** option is activated. The default screen name from the application is retrieved. Click **Clear** to undo the capture.
 - d. Click **Advanced Settings** to perform the task only when a certain condition is satisfied. For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.
11. After you capture all the application fields, click **Next**.
 12. In the Identify Screen for Logon window, provide one or more case-sensitive text strings that can accurately identify the screen.
 13. Click the **Add** button.
 14. After you add all the case-sensitive strings, click **Next**.
 15. Select each action to include for logging on in the **Available actions** group and moving it to the **Selected actions** group.

Important: The actions must follow the correct sequence for the logon process.

- a. Select actions from the **Available actions** pane.
- b. Click the **Add** button.
- c. Use the **Move Up** and the **Move Down** icons to arrange the sequence of actions in the **Selected actions** pane.

16. After you select the sequence of actions, click **Next**.
17. Specify whether you want AccessStudio to identify the successful completion of the logon task.
 - The options are **No, continue without identifying successful logon** (no success screen or message displays) and **Yes, find the screen that appears upon successful logon** (capture the success screen of the application).
 - If you selected **Yes, find the screen that appears upon successful logon**, provide one or more case-sensitive text strings that can accurately identify the screen. Click **Add**.
18. After you add all the case-sensitive strings, click **Next**.
19. Select or create an authentication service.

If there was no previously created authentication service, leave the default selection at **Create one for me automatically**.
20. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

21. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile, and select **Upload to IMS**.
 - b. Click **Yes** when the IMS Upload Confirmation is displayed.

The system displays a message, indicating the success or failure of the upload.

What to do next

If you must capture multiple AccessProfiles for a single Mainframe emulator, create advanced AccessProfiles.

For more information, see “Creating advanced AccessProfiles” on page 51.

Creating a change password AccessProfile for Mainframe or cursor-based applications

Use the AccessProfile Generator to create a change password AccessProfile for a Mainframe or cursor-based application.

Procedure

1. Follow the steps from Open AccessStudio to Enter a unique name for the application of the “Creating a logon AccessProfile for Mainframe or cursor-based applications” on page 34 procedure.
2. Specify the task to automate by selecting the appropriate radio button. For this procedure, select **Change password**.
3. Click **Next**.
4. Enter a unique name for the screen.
5. Based on your selected task, capture identification information for the fields on the application screen.
 - a. Click the **Finder tool** from the AccessProfile Generator.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. Drag the **Finder tool** to the corresponding field in the application screen. As you drag the **Finder tool** to the application, the AccessProfile Generator marks the field or button that can be captured.
 - c. When the **Finder tool** is positioned over the field, release the mouse button. If a field was captured successfully, the **Clear** option is activated. The default screen name from the application is retrieved. Click **Clear** to undo the capture.
 - d. Click **Advanced Settings** to perform the task only when a certain condition is satisfied. For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.
6. After you capture all the application fields, click **Next**.
 7. In the Specify Text to Identify Change Password Screen window, provide one or more case-sensitive text strings that can accurately identify the screen.
 8. Click **Add**.
 9. After you add all the case-sensitive strings, click **Next**.
 10. Select each action to include for changing a password in the **Available actions** list and moving it to the **Selected actions** group.

Important: The actions must follow the correct sequence for the password change process.

- a. Select actions from the **Available actions** pane.
 - b. Click the **Add** button.
 - c. Use the **Move Up** and the **Move Down** icons to arrange the sequence of actions in the **Selected actions** pane.
11. Click **Next**.
 12. Specify whether you want AccessStudio to identify the successful completion of the password change task.
 - The options are **No, continue without identifying successful change** (no success screen or message displays) and **Yes, find the screen that appears upon successful change** (capture the success screen of the application).
 - If you selected **Yes, find the screen that appears upon successful change**, provide one or more case-sensitive text strings that can accurately identify the screen. Click **Add**.
 13. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

- a. In the Data type pane, right-click on the AccessProfile.
- b. Select **Upload to IMS**.
- c. Click **Yes** when the IMS Upload Confirmation is displayed.

The system displays a message, indicating the success or failure of the upload.

Creating a logoff AccessProfile for Mainframe or cursor-based applications

Use the AccessProfile Generator to create a logoff AccessProfile for a Mainframe or cursor-based application.

Procedure

1. Follow the steps from Open AccessStudio to Enter a unique name for the application of the “Creating a logon AccessProfile for Mainframe or cursor-based applications” on page 34 procedure.
2. Specify the task that you want to automate by selecting the appropriate option. For this procedure, select **Logoff**.
3. Click **Next**.
4. Enter a unique name for the screen.
5. Specify whether the logoff screen is displayed as a Windows dialog by selecting the appropriate radio button.

If you select **Screen is not a pop-up dialog box**, do the following tasks:

- a. Click the **Finder tool** from the AccessProfile Generator, then drag the icon on the Windows dialog box. When the **Finder tool** is over the dialog, release the mouse button.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. To identify the screen title, click a **Finder tool** from the AccessProfile Generator, then drag the icon on the matching field in the application screen.
- c. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear signature** option is activated.
- d. The **Screen title** field retrieves its default screen name from the application. Click **Clear signature** to undo the capture.
- e. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

- f. Proceed to the next step.

If you select **Screen is a pop-up dialog box**, do the following tasks:

- a. Click the **Finder tool** from the AccessProfile Generator, then drag the icon on the Windows dialog box. When the **Finder tool** is over the dialog, release the mouse button.

Note: If the **Finder tool** is not enabled, click the **Edit Signature** link, then close the balloon that opens, which enables the **Finder tool**.

- b. To identify the screen title, click a **Finder tool** from the AccessProfile Generator, then drag the icon on the matching field in the application screen.
- c. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear signature** option is activated.
- d. The **Screen title** field retrieves its default screen name from the application. Click **Clear signature** to undo the capture.
- e. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

- f. Proceed to Selecting actions for logging off step.
6. After you capture all the application fields, click **Next**.
7. In the Identify Screen for Logoff window, provide one or more case-sensitive text strings that can accurately identify the screen.
8. Click **Add**.
9. After you add all the case-sensitive strings, click **Next**.
10. Select each action to include for logging off in the **Available actions** list and moving it to the **Selected actions** group.

Important: The actions must follow the correct sequence for the logoff process.

- a. Select actions from the **Available actions** pane.
- b. Click the **Add** button.
- c. Use the **Move Up** and the **Move Down** icons to arrange the sequence of actions in the **Selected actions** pane.
11. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload it to the IMS Server. For more information, see Chapter 6, "AccessProfiles testing," on page 113.

12. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile.
 - b. Select **Upload to IMS**.
 - c. Click **Yes** when the IMS Upload Confirmation is displayed.

The system displays a message, indicating the success or failure of the upload.

Creating logon AccessProfiles for Mainframe applications with HLLAPI support

Use the AccessProfile Generator for set logon profiles for Mainframe applications with HLLAPI support.

About this task

High Level Language Application Programming Interface (HLLAPI) is a standard to access existing information and applications that are housed on IBM and Unisys Mainframes, AS/400, UNIX/VMS, and others.

Examples of applications that provide HLLAPI are Attachmate EXTRA, Reflection, and IBM iSeries. Terminal Emulators are typically used to connect to these existing applications.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Click **Next** from the Welcome window.
4. Open the application logon screen that requires an AccessProfile.
5. Click **Next**.
6. Enter a unique name for the application and select **Mainframe application with HLLAPI support** as the application type.

7. Click **Next**.
8. In the Select Task to Automate screen, **Logon** is the only available option.
9. Click **Next**.
10. Provide the necessary HLLAPI information.

All installations of Terminal Emulators must have the following settings.

The configuration varies for each application. See the documentation of the specific application for configuration information.

Option	Description
DLL filename	Enter the Dynamic Linked Library for HLLAPI functionality that is provided by the Terminal Emulator Application, or click Browse to navigate to the .DLL file stored on the computer.
Short Name	Select a unique name to identify a host session. Select a letter from A to Z. This field is mandatory.
Long Name	Select an alternate name to identify a host session, up to eight characters.

11. In the Identify Screen for Logon window, enter a unique name for the screen.
12. Based on your selected task, capture identification information for the fields on the application screen.

Note: The fields available for each automated task varies.

- a. Click the **Finder tool** from the AccessProfile Generator, then drag the icon on the matching field in the application screen or web page.
- b. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured.
- c. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear signature** option is activated.
- d. The screen title field retrieves its default screen name from the application. Click **Clear signature** to undo the capture.
- e. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

13. After you capture all the application fields, click **Next**.
14. In the Identify Screen for Logon window, provide case-sensitive text strings that can accurately identify the screen. Click **Add**.
15. Click **Next**.
16. Select each action to include for logging on in the **Available actions** group and moving it to the **Selected actions** group.

Important: The actions must follow the correct sequence for the logon process.

- a. Select actions from the **Available actions** pane.
- b. Click the **Add** button.
- c. Use the **Move Up** and the **Move Down** icons to arrange the sequence of actions in the **Selected actions** pane.

17. Specify whether you want AccessStudio to identify the successful completion of the logon task.
 - The options are **No, continue without identifying successful logon**, and **Yes, find the screen that appears upon successful logon**.
 - If you selected **Yes**, find the screen that displays, key in one or more case-sensitive text strings that appear on the screen after successful logon. Then, click **Add**.
18. Click **Next**.
19. Select or create an authentication service.

If there was no previously created authentication service, leave the default selection at **Create one for me automatically**.
20. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before you upload it to the IMS Server. For more information, see Chapter 6, "AccessProfiles testing," on page 113.

21. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile.
 - b. Select **Upload to IMS**.
 - c. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

Creating logon AccessProfiles for other applications

Use the AccessProfile Generator for set logon profiles for other applications.

About this task

For other applications, logon AccessProfiles can be created only by using AccessStudio.

Procedure

1. Open AccessStudio (**Start > All Programs > ISAM ESSO AccessStudio > AccessStudio**).
2. Start the AccessProfile Generator by clicking the **New** icon and selecting **New AccessProfile** (using Generator).
3. Click **Next** from the Welcome window.
4. Open the application logon screen that requires an AccessProfile.
5. When you open the application screen or web page, click **Next**.
6. Enter a unique name for the application and select **Other applications** as the application type.
7. Click **Next**.
8. Specify the task that you want to automate by selecting the appropriate radio button.
9. Click **Next**. The **Logon** option is selected by default.
10. Enter a unique name for the screen or web page to capture in the Identity Screen for Logon window.
11. Based on your selected task, capture identification information for the fields on the application screen.

Note: The fields available for each automated task varies.

- a. Click the **Finder tool** from the AccessProfile Generator, then drag the icon on the matching field in the application screen or web page. As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. You can also add unique screen text for identification.
- b. When the **Finder tool** is positioned over the field, release the mouse button. If the field was captured successfully, the **Clear signature** option is activated.
- c. The **screen title** field retrieves its default screen name from the application. Click **Clear signature** to undo the capture.
- d. Click **Advanced settings** to perform the task only when a certain condition is satisfied.

For more information, see “Creating AccessProfiles that perform automation tasks” on page 42 or “Setting advanced options” on page 50.

12. After capturing all the application fields, click **Next**.
13. Select each **Action** to include for logging on in the **Available actions** group and moving it to the **Selected actions** group.
 - a. Select each action, then click **Add**.
 - b. Arrange the order of the selected actions by using the **Move Up** and **Move Down** icons.

Important: The actions must follow the correct sequence for the logon process.

14. Select or create an authentication service.
If there was no previously created authentication service, leave the default selection at **Create one for me automatically**.
15. Click **Finish** to return to the AccessStudio user interface.

Important: Test all the AccessProfiles before uploading to the IMS Server. For more information, see Chapter 6, “AccessProfiles testing,” on page 113.

16. Upload the AccessProfile to the IMS Server to activate it.
 - a. In the Data type pane, right-click on the AccessProfile.
 - b. Select **Upload to IMS**.
 - c. Click **Yes** when the IMS Upload Confirmation displays.

Another message box displays, indicating the success or failure of the upload.

What to do next

The standard AccessProfile diagram in the **States** tab is not enabled. It changes color when you click the **Enable state editing** link. In this case, you can edit and convert your standard AccessProfiles into advanced AccessProfiles.

Automating tasks with AccessProfiles

The AccessProfile Generator wizard supports the creation of AccessProfiles that automate tasks which are not related to application logon, password change, or logoff.

Examples of automation tasks are:

- Clicking a button when a screen is displayed.
- Navigating a Web application to a particular page after logon.

- Submitting a custom audit log.
- Running a VBScript.

For more information, see the following topics:

- “Creating AccessProfiles that perform automation tasks”
- “Clicking controls” on page 43
- “Disabling controls” on page 43
- “Enabling controls” on page 44
- “Hiding controls” on page 44
- “Entering text” on page 45
- “Pausing actions” on page 45
- “Pressing keys” on page 46
- “Running plug-ins” on page 46
- “Closing screens” on page 47
- “Selecting menu items” on page 47
- “Generating custom audit logs” on page 48

Creating AccessProfiles that perform automation tasks

Use AccessStudio to create AccessProfiles for your automation tasks.

Procedure

1. Select **New icon > New Access Profile (Using Generator)** from the AccessStudio menu. The AccessProfile Generator wizard is displayed.
 2. Click **Next**.
 3. Select **Windows (Win32, 16bit) application** (for example) as the application type.
 4. Click **Next**.
 5. Select **Other Tasks** from the **Select Task to Automate** list.
 6. Click **Next**. The Identify Screen for Other Tasks window is displayed.
 7. Drag the **Finder tool** and drop it to the application window or a web page that has a task to automate.
 8. If you want AccessAgent to perform the task only when a certain condition is satisfied, click **Advanced Settings**. The Advanced Settings window is displayed.
Otherwise, proceed to the Click Next to go to the Specifications for Other Task window step.
- Note:** The **Advanced Settings** button is only activated after you capturing the necessary field information from the application.
9. Select the **Set condition** link. The AccessAgent Plug-in Editor window is displayed.
 10. Replace the two lines of commented text: Define function body here and do NOT modify the function template with your own VBScript or JScript.
If the function returns true, the condition is satisfied. For more information, see “Using plug-in API for customized triggers” on page 87.
 11. Click **OK** to return to the Advanced Settings window.
 12. Click **OK** again to return to the Identify Screen for Other Task window.
 13. Click **Next** to go to the Specifications for Other Task window.

14. Select a typical automation task to be performed from the **Available actions** list.

What to do next

After you complete this procedure, you can set up an automation task.

See the following procedures for instructions on each type of automation task.

- “Clicking controls”
- “Disabling controls”
- “Enabling controls” on page 44
- “Hiding controls” on page 44
- “Entering text” on page 45
- “Pausing actions” on page 45
- “Pressing keys” on page 46
- “Running plug-ins” on page 46
- “Closing screens” on page 47
- “Selecting menu items” on page 47
- “Generating custom audit logs” on page 48

Clicking controls

Use AccessStudio to create an AccessProfile that automates a task, such as clicking a user interface control element.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Click a control**.
2. From the **Additional configuration** field, drag the **Finder tool** over the target control button, field, or link.

As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. When the **Finder tool** is positioned over the field, release the mouse button.

3. Click **Add** from the **Selected actions** field.
4. Click **Next**.
5. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click the **Delete** button.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Disabling controls

Use AccessStudio to create an AccessProfile that automates a task, such as not enabling a user interface control element.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Disable a control**.
2. From the **Additional configuration** field, drag the **Finder tool** over the target control button, field, or link to not enable.
As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. When the **Finder tool** is positioned over the field, release the mouse button.
3. Click **Add** from the **Selected actions** field.
4. Click **Next**.
5. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Enabling controls

Use AccessStudio to create an AccessProfile that automates a task, such as enabling a user interface control element.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Enable a control**.
2. From the **Additional configuration** field, drag the **Finder tool** over the target control button, field, or link to enable.
As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. When the **Finder tool** is positioned over the field, release the mouse button.
3. Click **Add** from the **Selected actions** field.
4. Click **Next**.
5. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Hiding controls

Use AccessStudio to create an AccessProfile that automates a task, such as removing or hiding a user interface control element.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Hide a control**.
2. From the **Additional configuration** field, drag the **Finder tool** over the target control button, field, or link to hide.
As you drag the **Finder tool** to the application, AccessProfile Generator marks the field or button that can be captured. When the **Finder tool** is positioned over the field, release the mouse button.
3. Click **Add** from the **Selected actions** field.
4. Click **Next**.
5. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Entering text

Use AccessStudio to create an AccessProfile that automates a task, such as automatically entering text for specific fields.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Enter text**.
2. Enter the text that you want to be sent to the target control when the task is run.
3. Click **Add** from the **Selected actions** field.
4. Click **Next**.
5. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Pausing actions

Use AccessStudio to create an AccessProfile that automates a task, such as setting a time delay for a task.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions**, select **Pause**.
2. From the **Additional configuration, Time (in secs)** field, enter the time delay of the pause while running the selected task.
3. Click **Add** from the **Selected actions** field.
4. Click **Next**.
5. From the **Screens identified** field, select the captured screen for the task to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Pressing keys

Use AccessStudio to create an AccessProfile that automates a task, such as sending keystrokes to an application.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Press key**.
2. Choose the **Press key** option to send certain keystrokes to the application.
3. Click **Add** from the **Selected actions** field.
4. Select the check box of the key or a key combination from the **Additional configuration** box.
5. Click **Next**.
6. From the **Screens identified** field, select the captured screen for the task to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Running plug-ins

Use AccessStudio to create an AccessProfile that automates a task, such as running plug-in scripts.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Run plug-in**.
2. Choose this option to perform a customized action.
3. Click the **Specify plug-in script** link from the **Additional configuration** box. The AccessAgent Plug-in Editor is displayed.

4. Replace the two lines of commented text Define function body here and do NOT modify the function template with your own VBScript or JScript. Click **OK**.

If the function returns true, the condition is satisfied. For more information, see “Using advanced AccessProfiles to meet custom requirements” on page 71.

5. Click **Add** from the **Selected actions** field.
6. Click **Next**.
7. From the **Screens identified** field, select the captured screen for the task to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Closing screens

Use AccessStudio to create an AccessProfile that automates a task, such as automatically closing screens or windows.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Close the screen**.
2. Click **Add** from the **Selected actions** field.
3. Click **Next**.
4. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Selecting menu items

Use AccessStudio to create an AccessProfile that automates a task, such as automatically selecting menu items for a screen or window.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Select menu item**.
2. From the **Additional configuration** field, drag the **Finder tool** over the target menu item.

As you drag the **Finder tool** to the application, AccessProfile Generator marks the menu item that can be captured.
3. Enter the menu path.
4. Click **Add** from the **Selected actions** field.
5. Click **Next**.

6. From the **Screens identified** field, select the captured screen for the task you want to automate.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Generating custom audit logs

Use AccessStudio to create an AccessProfile that automates a task, such as generating audit log entries for specific events. The audit log is sent to the IMS Server.

Before you begin

Complete the procedure in “Creating AccessProfiles that perform automation tasks” on page 42.

Procedure

1. From the **Available actions** list, select **Generate custom audit log**.
2. Click **Specify log properties** link from the **Additional configuration** box. The Custom Audit Log Action window is displayed.
3. Select an event code from the list.

Note: To create custom events, go to **AccessAdmin > System policies > AccessAudit Policies**.

4. Select a result code from the list.
5. Specify a name in the Available name-value pairs pane. Then, select a value from the list.

Each name-value pair that is specified here displays in the audit log entry that is submitted to the IMS Server when the event occurs.
6. In the Specify value pane, drag the **Finder tool** and drop it to the field that you must capture as the value is the name-value pair.
7. Click **Add** from the Name-value pairs to be logged pane.
8. Click **OK** to return to the Specify Actions for Other Task pane.
9. Click **Add** to populate the **Selected actions** field.
 - Click **Next** to go to the Identify Screens and Fields for Other Tasks window.
 - To delete the screen, select the screen name and click **Delete**.
 - Click **Add** to add more screens.
 - Click **Finish** to return to the AccessStudio user interface.

Editing standard AccessProfiles

When you have created standard AccessProfiles using the AccessProfile Generator, you still can add, delete, or edit tasks in the **General Properties** tab of AccessStudio.

See the following procedures for more information:

- “Adding tasks” on page 49
- “Deleting tasks” on page 49
- “Editing tasks” on page 49
- “Setting advanced options” on page 50

Adding tasks

Use AccessStudio to create a task to automate for your application.

Procedure

1. Select **Add** to go to the Select Task to Automate window.
2. Select a task. For example, Log off.
3. Click **Next**.
4. In the Identify Application signature, drag the crosshair into the application that you want to log off.
5. Click **Next**.
6. Select a method for automatic Logoff. For example, Log off gracefully.
7. Click **Next**.
8. Select a **Graceful Logoff** option.
9. Click **Next**. The Identify Screen for Logoff window is displayed.
10. Drag the **Finder tool** and drop it to the logoff button or field.
11. Click **Next**.
12. Select an action from the **Available actions** drop-down list.
13. Click **Add**.
14. Click **Next**. The Identify Screen for Logoff window is displayed.
15. Select **Logoff screen 1** from the **Screens (last screen if checked)** field.
16. Click **Finish**. The **General Properties** tab is displayed.

Deleting tasks

Use AccessStudio to delete a previously automated task in your application.

Procedure

1. Select the task to delete from the Tasks captured pane.
2. Click **Delete**.

Editing tasks

Use AccessStudio to modify information in an automated task that is used for your application.

Procedure

1. Select the task to edit from the Tasks captured pane. For example, Logon.
2. Click **Edit** to go to the Identify Screens and Fields for Logon window.
3. From the Screens identified pane, select a screen. Click **Edit** to go to the Identify Screen and Fields for Logon window.
4. Click **Clear signature** to recapture a field.
5. Drag the **Finder tool** and drop it to the target field or button.
You can also click **Edit signature** to make the necessary changes.
6. Click **Next** to return to the Identify Screen and Fields for Logon window.
7. Click **Next** to go to the Identify Successful Logon window.
8. Edit the title or text on the **Screen title** field.
9. Click **Next** to go to the Select or create authentication service window.
10. Select or create an authentication service.
11. Click **Finish** to return to the **General Properties** tab.

Setting advanced options

Use AccessStudio to set advanced options for the automated task used for your application.

Procedure

1. In the Identify Screen and Fields for Logon window, click **Advanced settings**.
2. Edit the signature in the **Screen signature** field.
3. In the Injection delay pane, the auto-fill delay is set to 0 by default.
If auto-fill fails, specify the length of the delay before an auto-fill.
4. Click the **Set condition** link to set conditions for customized triggers. The AccessAgent Plug-in Editor window is displayed.
5. In the **Script** pane, replace the two lines of commented text Define function body here and do NOT modify the function template with your own VBScript or JScript.
If the function returns true, the condition is satisfied. For more information, see “Using advanced AccessProfiles to meet custom requirements” on page 71.
6. Click **OK**.

Chapter 4. Advanced AccessProfiles

The AccessProfile Generator automatically generates an AccessProfile, but it does not correspond to a state-engine script or advanced AccessProfile.

To convert AccessProfiles into a state-engine script, go to the **States tab diagram** and click the **Enable state editing** link on the upper left corner. You can edit the states, triggers, actions, and their properties.

You can also use the **New** icon to create advanced AccessProfile.

For more information, see the following topics:

- “Creating advanced AccessProfiles”
- “Editing advanced AccessProfiles” on page 53
- “Creating sample advanced AccessProfiles” on page 54
- “Checking and prompting for relogin” on page 69
- “Adding fields for random passwords” on page 70
- “Using advanced AccessProfiles to meet custom requirements” on page 71

To learn more about creating Advanced AccessProfiles, go to the IBM Redbooks® web page and search for AccessProfiles.

Creating advanced AccessProfiles

Use the **General Properties** tab to create an advanced AccessProfile (State Engine SSO Support).

Procedure

1. Add an AccessProfile.
 - a. Click the **New** icon and select **New Advanced AccessProfile**. An AccessProfile is automatically created with a default name of `profile_1`, which you can modify from the Details pane.
 - b. In the **General Properties** tab, enter the name of the AccessProfile you created.
 - c. Click **Add** from the **Signatures** field. The Signature window is displayed.
 - d. Drag the **Finder tool** and drop it to the field or window of the application which needs an AccessProfile. The site signature is generated.
 - e. Click **OK**.

This action takes you back to the **General Properties** tab.

2. Specify the extra support information, depending on the application type.
 - a. Click the **Extra Support Information** twistie.
 - b. Select the corresponding check box of your chosen extra support information.

You can choose to enable screen scraping support for terminal and mainframe applications; HLLAPI support for mainframe applications; and support for Java applications.

3. Optional: Select the application object.

AccessStudio automatically creates an application object for every AccessProfile.

You can also link the AccessProfile to a different application instead of the default new one.

- a. Click the **Advanced Options** twistie to expand.
 - b. Select an application object from the **Application Id** list. If the application object is not yet created, see “Creating an application object” on page 127 for instructions to create it.
4. Add states.
- a. Select the **States** tab. Select the **New state** icon from the menu to add another state. A new state is created.
 - b. Specify the state information in the Properties pane.
5. Add triggers.
- a. Select the state where you must add the trigger.
 - Right-click on the state or click the **New trigger** icon from the menu.
 - Select **Add Trigger**.
 - Select the appropriate trigger from the list.A new trigger is created.
 - b. Specify the trigger information in the **Properties** tab.

A signature is unique identification information for any window or field. For some triggers, you must capture the signature of an associated window.
 - c. To capture a signature, click **Edit** next to a field for capturing a signature to display the Signature window.

Drag the **Finder tool** and drop it to the field or window of the application which needs an AccessProfile. The signature is automatically generated and displayed in the **Generated Signature** text box.
 - d. Optional: Click **Description** to open a text field where you can enter a note or description about the trigger.
6. Add actions.
- a. When you specify the trigger information, add actions to triggers.
 - 1) Select the trigger.
 - 2) Right-click on the trigger or click the **New action** icon from the menu.
 - 3) Select **Add action** and select the appropriate action from the list.
 - 4) Specify the action information in the **Properties** tab.
 - b. Optional: Click **Description** to open a text field where you can enter a note or description about the action.
 - c. Script your own action if the action you require is not listed.

For more information, see “Customizing actions” on page 72.

Capture the signature of an associated window for some actions. Follow instructions in To capture a signature step.
7. Capture identification information for the logon fields on the application screen.
- a. Specify credential information for capture by using SSO-items for some actions.

AccessAgent captures or auto-fills credential information from fields on a screen called SSO-items. Example: The user name or password field on a window. SSO-items correspond to info-controls in a Standard AccessProfile. Configure SSO-items to include the format of information to be captured (for example, if you are creating an AccessProfile for the *Messaging Software*,

and all users in your organization are required to have email software accounts, create two actions for capturing and auto-filling the information).

- b. In the first action, specify that when capturing information from a *Messaging Software*, AccessAgent ignores the @example.com and capture only the user name.

Then, in the second action you can specify that AccessAgent appends a captured *Messaging Software* user name with @example.com for all users. Test all the AccessProfiles before uploading to the IMS Server. For more information, see “Testing AccessProfiles” on page 113.

8. Upload the AccessProfile to IMS Server.

- a. Upload the AccessProfile you created to the IMS Server connected in AccessAgent. This action activates the AccessProfile.
- b. Right-click on the AccessProfile and click AccessProfiles to the IMS Server. The message that is displayed indicates the upload success or failure.

Note: You can also select and upload multiple AccessProfiles simultaneously.

Editing advanced AccessProfiles

Use AccessStudio to change information for your advanced AccessProfiles.

About this task

You can change the AccessProfile information by using either of the following AccessStudio elements:

- States diagram
- General Properties tab
- XML Editor tab

Note: The **XML Editor** tab is for advanced functionality editing, and can be used only by more advanced AccessStudio users.

Editing advanced AccessProfiles from the States diagram

Use the **States diagram** to modify information for your advanced AccessProfiles.

Procedure

1. Select the AccessProfile from the Data type pane. The **States** tab on the Details pane is automatically displayed.
2. In the **States diagram**, select the state, trigger, or action to edit.

Tip: If your state diagram is too large and you cannot see the states, triggers, or actions on the other end, do any of the following tasks:

- Select the state, trigger, or action, then press **F9** or **Shift + F9** on your keyboard to move to the next state, trigger, or action until you see what you must edit.
 - Press the left mouse button on an empty area of the state diagram canvass until you see a hand cursor. Then, drag the state diagram canvass to the area you must edit.
3. Right-click and select the appropriate option from the menu.
You can also click the corresponding icon on the **States diagram** toolbar. See the States diagram toolbar section of for details.
 4. Select the state to edit the state transitions.

5. Go to the **Properties pane > Form Editor tab** to indicate the next state ID.
6. Configure the rest of the necessary fields in the **Properties pane > Form Editor tab**.

Editing advanced AccessProfiles from the General Properties tab

Use the **General Properties** tab to modify information for your advanced AccessProfiles.

Procedure

1. Select the list item from the Data pane. The **General Properties** tab on the Details pane is automatically displayed.
2. Click **Edit** from the Signatures identifying web page or .EXE where this profile is to be loaded to start editing. The Signature window is displayed.
3. Select the appropriate option from the **Signature for** list.
4. Edit the signature from the **Generated Signature** field.
5. Select the check box based on what you must generate.
6. Click **OK** to return to the **General Properties** tab.
7. Optional: Click **Description** to enter a note or description about the trigger.

Editing advanced AccessProfiles from the XML Editor tab

Use the **XML Editor** tab to modify information for your advanced AccessProfiles.

Procedure

1. Modify the XML script that you must edit.
2. Right-click to undo, cut, copy, paste, delete, or to select all.

Note: AccessStudio advanced users can also click the **Replace Text** icon in the States tab toolbar to find and replace XML data in your XML Editor.

Creating sample advanced AccessProfiles

For more information about setting up advanced AccessProfiles, see these examples of advanced AccessProfiles.

Login scenario

Use AccessStudio to set up an automatic logon AccessProfile.

About this task

Create an AccessProfile to enable automatic logon for the Logon window of the sample application, Patient Information Manager (PIM).

Procedure

1. Launch AccessStudio.
2. Click the **New** icon.
3. Select **New Advanced AccessProfile**. A **new profile1** data item is created.
4. In the **General Properties** tab at the Details pane, specify a different name for the AccessProfile in the **Id** field.
5. Capture the information that identifies the application through a signature.

- See “Capturing the signature” on page 56 for detailed steps.
6. Add an application object to the AccessProfile.
See “Creating an application object” on page 56 for detailed steps.
 7. Select **AccessProfiles** from the **View** menu.
 8. Select **adv_ap_wnd_patientinformationmanager_v1** from the Data type pane.

Example

The following diagram illustrates the states, triggers, and actions to create to enable automatic logon for this application.

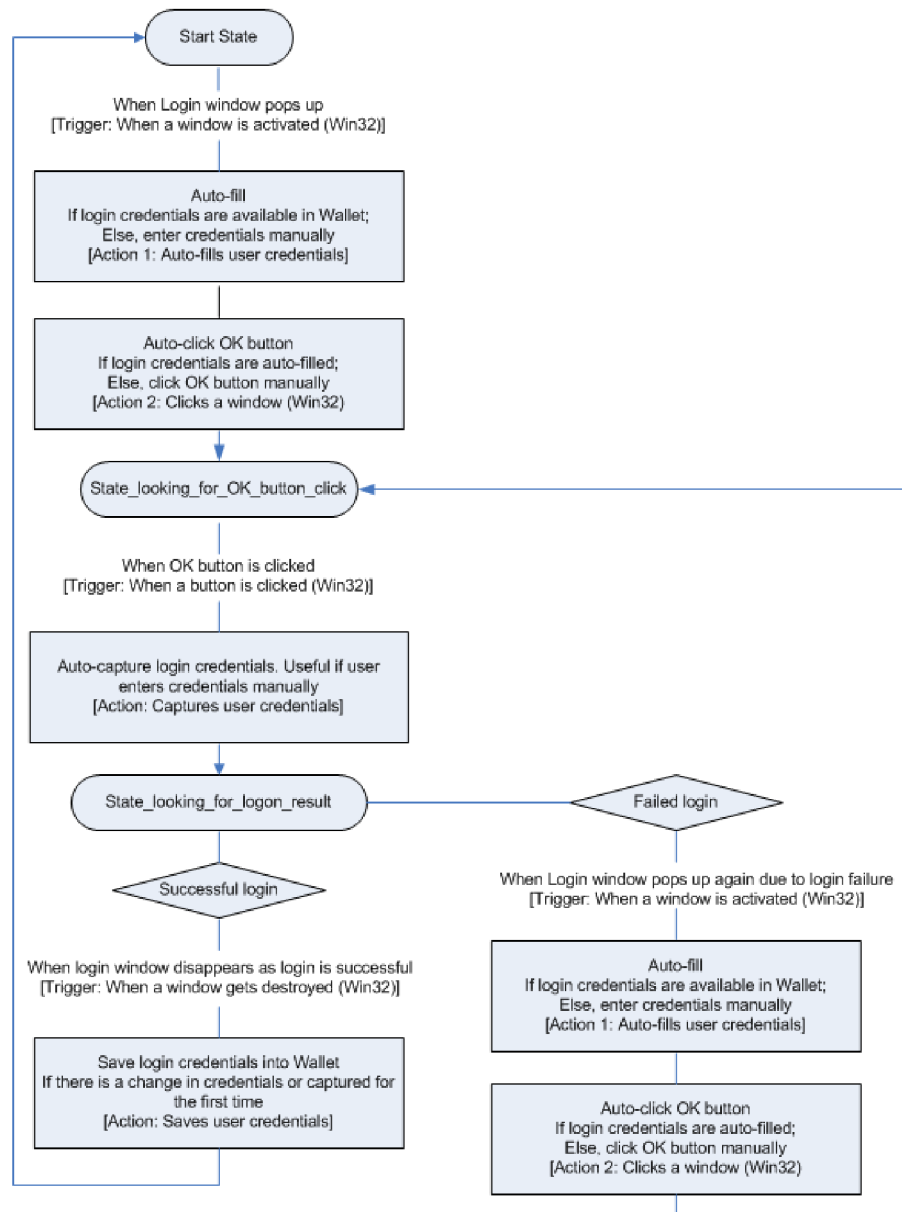


Figure 4. Sample workflow to enable automatic logon

What to do next

See the following topics for details on creating the Login scenario.

- “Making the state engine work” on page 61
- “Creating the state engine” on page 57
- “Validating the logon workflow” on page 62
- “Change password scenario” on page 63

Capturing the signature

Use AccessStudio to capture the signature of the sample advanced AccessProfile.

Procedure

1. Click **Add** in the **Signatures identifying web-page or exe where this is AccessProfile is to be loaded** field.

The Signature Window is displayed.

The main AccessStudio window is hidden when the Signature Window is displayed. If the Patient Information Manager application is not displayed, click the **Signature window task** tab in the taskbar to display the application window.

Note: Select the **Hide this window while dragging the finder tool** check box if you do not want to see the Signature Window while capturing the signature. Otherwise, leave the check box cleared. Select the appropriate check boxes at the bottom of the Signature Window if you want to generate the .EXE file version and the company name in the signature.

2. Drag the **Finder tool** and drop it on the main Patient Information Manager application window. The signature of the Patient Information Manager application is captured and displayed in the **Generated Signature** field.

If you marked both the check boxes at the bottom of the Signature Window, the captured signature is displayed as:

```
/child::exe[@exe_name="PatientInfo.exe" and  
@company_name="IBM" and  
@file_version="1.0.2307.30260"]
```

Note: You can edit the signature in the field. For example, remove the `and @file_version="1.0.2307.30260"` portion to enable the AccessProfile to work for all versions of the Patient Information Manager application. However, to differentiate between versions, be sure to retain this portion.

3. Click **OK** to return to the main AccessStudio screen.

Creating an application object

Use AccessStudio to create application objects for an advanced AccessProfile.

Procedure

1. Select **New > New Application** from the menu.
2. Modify the details in the corresponding fields on the **General Properties** tab.
3. Modify the application ID if necessary. For example, enter `app_PIM` in the **Id** field.
4. Enter the application name in the **Name** field. For example, `PatientInformationManager`.
5. Optional: Enter description information, if necessary.

6. Click the **Policies** to expand the category.
7. Specify an action for the application when the user logs off from AccessAgent.
8. Specify the default automatic sign-on option.

Creating the state engine

Use AccessStudio to create the state engine for the Patient Information Login scenario.

Procedure

1. Create an advanced AccessProfile for the Patient Information Login scenario.
 - a. Select the **Logon_profile1** you created earlier from the AccessProfiles pane.
 - b. Select the **States** tab. The States diagram is displayed.
 - c. Select **Start State** from the States diagram.
2. Create a **Window is activated** trigger.

This trigger fires when a specific window is activated.

Right-click on the Start_State, select **Add Trigger > Window is activated**.
3. Create a state `state_looking_for_OK_button_click` state.

This state enables the capturing of the user credentials to save in the Wallet for future auto-filling.

 - a. Click the **New state** icon from the States toolbar.
 - b. Select **New State 1**.
 - c. In the **Properties pane > Form Editor tab**, enter `state_looking_for_OK_button_click` in the **State Name** field.
4. Configure the **Window is activated** trigger.
 - a. Select the **Window is activated** trigger.
 - b. In the **Properties pane > Form Editor tab**, select `state_looking_for_OK_button_click` from the **Next state id** list.
 - c. Capture the Logon window signature.

Click **Edit** below the **Signature of the window getting the activate msg** field.

Drag the **Finder tool** and drop it on the Login window.

The generated signature is:

```
/child::wnd[@title="Login" and @class_name="#32770"]
```
 - d. Click **OK**.
5. Create an **Inject credentials** action for the **Window is activated** trigger. The action automatically inserts user credentials in the relevant fields.
6. Right-click on the **Window is activated** trigger.
7. Select **Add Action > Inject credentials**.
8. Configure the **Inject credentials** action.

The **default_injection_bag** option from the **Account data bag id** list is automatically selected.

An account data bag is a temporary virtual holder that stores the account data (user credentials) at the point of credential auto-fill, or capture.

 - a. Click the **Injection fields** twistie to expand.
 - b. Click **Add** next to the **Injection fields** list.
 - c. Select `aditi_ciuser` from the **Account data item template id** list.

This setting indicates that the field is not case-sensitive and a user name field.

This setting corresponds to the nature of credentials that are specified for the authentication service.

- d. Capture the signature of the **User name** field.

Click **Edit** below the **Signature of the window for injection or capture** field.

Drag the **Finder tool** and drop it on the **User name** field. Click **OK**.

The generated signature is:

```
/child::wnd[@title="Login" and  
@class_name="#32770"]/child::wnd[@class_name="Edit" and  
@ctrl_id=201]
```

- e. Create another **Windows control SSO item** for the **Password** field.

Capture the signature for the **Password** field.

- 1) Click **Add** beside the **Capture fields** list.
- 2) Select **aditi_cspwd** from the **Account data item template id** list.

This setting indicates that the field is a case-sensitive password field, thus making it a secret field.

- 3) Click **Edit** below the **Signature of the window for injection or capture** field.
- 4) Drag the **Finder tool** and drop it on the **Password** field.

The generated signature is:

```
/child::wnd[@title="Login" and  
@class_name="#32770"]/child::wnd[@class_name="Edit" and  
@ctrl_id=202]
```

- 5) Click **OK**.

- f. Add an authentication service to the **Inject credentials** action.

Since you are creating an AccessProfile for the first time, create an authentication service to associate with the specific window of the application.

- 1) Go to **View > Authentication Services**.
- 2) Click the **New** icon from the menu and select **New Authentication Service**.

This task displays the corresponding fields on the **General Properties** tab.

- 3) Enter **auth_PIM** in the **Id** field.
- 4) Enter **PIM** in the **Display Name** field.

Notice that the **Account data template id** by default is **adt_ciuser_cspwd**.

This setting indicates the nature of user credentials as a not case-sensitive user name and a case-sensitive password. Retain this selection.

For more information, see Chapter 10, "Account data items and templates," on page 131.

- 5) Return to the AccessProfiles function by selecting AccessProfiles in the **View** menu.
- 6) In the **States tab > Properties pane > Form Editor tab**, expand the **Auth Info** twistie.

- 7) In the **Add Auth Info** field, **Direct - Authentication Service** is selected by default. Click **Add** to expand the **Direct - Authentication Service** field.
- 8) Select **auth_PIM** from the **Authentication service id** list.
9. Create a **Click a window** action for the **Window is activated** trigger. This action enables automatic clicking of **OK** when the user name and password are automatically filled.

Note: The action happens only if the user selects **Auto-logout** for the authentication service.
10. Right-click on the **Window is activated** trigger.
11. Select **Add Action > Click a window** action.
12. Configure the **Click a window** action.
 - a. Select the **Click a window** action.
 - b. Capture the **OK** button signature.
 - 1) Click **Edit** below the **Signature of window to click** field.
 - 2) Drag the **Finder tool** and drop it on the **OK** button of the Login window.

The generated signature is:

```
/child::wnd[@title="Login" and @class_name="#32770"]/child::wnd[@class_name="Button" and @ctrl_id=1]
```
 - 3) Click **OK**.
 - 4) Click the **Advanced Options** twistie to expand.
 - 5) Select **Yes** from the **Execute only if autologon is enabled** list.
13. Create a **state_looking_for_logon_result** state. This state enables saving of credentials that are entered by the user.
 - a. Click the **New state** icon.
 - b. Select **New State 1**. Enter **state_looking_for_logon_result** in the **Properties pane > Form Editor tab > State Name** field.
14. Create a **Button is clicked** trigger for the **state_looking_for_OK_button_click** state. This trigger fires when a specific button is clicked a Windows application window. Right-click on the **state_looking_for_OK_button_click** state, select **Add Trigger > Button is clicked**.
15. Configure the **Button is clicked** trigger.
 - a. Select the **Button is clicked** trigger.
 - b. Select **state_looking_for_logon_result** from the **Properties pane > Form Editor tab > Next state id drop-down** list.
 - c. Capture the Login window signature. Click **Edit** below the **Signature of the window receiving the command notification** field. Drag the **Finder tool** and drop it to the Login window. The generated signature is:


```
/child::wnd[@title="Login" and @class_name="#32770"]
```
 - d. Click **OK**.
 - e. Capture the **OK** button signature.

Click **Edit** below the **Signature of control** field.

Drag the **Finder tool** and drop it on the **OK** button of the Login window.

The generated signature is:

```
/child::wnd[@title="Login" and  
@class_name="#32770"]/child::wnd[@class_name="Button" and  
@ctrl_id=1]
```

f. Click **OK**.

16. Create a **Capture credentials** action for the **Button is clicked** trigger.

Right-click on the **Button is clicked** trigger, select **Add action > Capture credentials**.

17. Configure the **Capture credentials** action.

The **default_capture_bag** is automatically selected in the **Account data bag id** list.

a. Click to expand the **Capture fields** twistie.

b. Click **Add** beside the **Capture fields** list.

c. Select **aditi_ciuser** from the **Account data item template id** list.

d. Capture the user name signature.

Click **Edit** below the **Signature of the window for injection or capture** field.

e. Drag the **Finder tool** and drop it on the **User name** field of the Login window.

The generated signature is:

```
/child::wnd[@title="Login" and  
@class_name="#32770"]/child::wnd[@class_name="Edit" and  
@ctrl_id=201]
```

f. Click **OK**.

g. Capture the **password** field signature.

1) Click **Add** next to the **Capture fields** list.

2) Select **aditi_cspwd** from the **Account data item template id** list.

3) Click **Edit** below the **Signature of the window for injection or capture** field.

4) Drag the **Finder tool** and drop it on the **Password** field on the Login window.

The generated signature is:

```
/child::wnd[@title="Login" and  
@class_name="#32770"]/child::wnd[@class_name="Edit" and  
@ctrl_id=202]
```

5) Click **OK**.

18. Specify the authentication service ID of the Patient Information Manager application.

a. Click to expand the **Auth Info** twistie.

b. Click **Add** beside the **Add Auth Info** list.

c. Select **auth_PIM** from the **Authentication service id** list.

19. Click the **Save selected data to file** icon from the **AccessStudio** toolbar to save the AccessProfile.

20. Select the **Upload selected data to IMS** icon from the **AccessStudio** toolbar to upload the AccessProfile in the IMS Server.

21. Click the **Start Test** icon from the **AccessStudio** toolbar to test the AccessProfile.

Making the state engine work

Use AccessStudio to check whether the state engine works for the sample application.

Procedure

1. In the advanced AccessProfile, define a trigger to fire associated actions to enable automatic-logon on the Login window of the Patient Information Manager application.

Use the **Window is activated** trigger. This trigger fires when a specific window is activated.

For the Patient Information Manager application, the automatic logon must occur when the Login window is displayed.

Use two actions:

- a. Auto-insert credential information in the logon fields (**Inject credentials**).
- b. Enable clicking of **OK (Click a window)**.

AccessAgent detects the Logon window, the credentials of the user is automatically inserted, and **OK** is clicked.

Note: In an alternate scenario, the user can manually enter credentials that were not previously saved in AccessAgent. In which case, the user manually clicks **OK**.

2. Define the `state_looking_for_OK_button_click` state.

3. Configure the AccessProfile to capture the entered credentials.

4. Add the **Button is clicked** trigger.

This trigger fires when a button is clicked. The transition to the next state occurs when **OK** is clicked (either automatically or by the user).

5. Save the captured credentials.

- a. Define the `state_looking_for_logon_result` state.

- b. Add two triggers to this state:

- If the logon is successful
- If the logon fails

When the logon credentials are captured, the transition to the `state_looking_for_logon_result` state occurs. The appropriate trigger fires based on the success or failure of the logon.

- c. Define the **Window gets destroyed** trigger for a successful logon.

This trigger fires when a window is deleted. For the Patient Information Manager application, when the logon is successful, the Logon window disappears, and the Patient Information Manager window is displayed to the user. The trigger fires when an event to close the Logon window occurs.

- d. Define the action (**Save credentials**) to save the logon credentials when this trigger fires.

For an auto-fill, the logon credential is not saved because they exist in the Wallet of the user.

- e. Define the **Window is activated** trigger again if logon fails.

For the Patient Information Manager application, the Logon window reappears if the logon fails.

- f. Define the two actions for these triggers again.

- **Inject credentials** action
- **Click a window** action

Results

A successful logon transitions to the start state. An unsuccessful logon transitions to the state_looking_for_OK_button_click state.

Validating the logon workflow

Use AccessStudio to check whether the logon AccessProfile is working properly.

Procedure

1. Open the **Logon** AccessProfile.
2. Select the **States** tab.
3. Create a **Window gets destroyed** trigger for the state_looking_for_logon_result state.

This trigger fires when a Windows application window is closed. The save credentials must be captured when the logon is successful.

Right-click on the state_looking_for_logon_result state and select **Add trigger > Advanced > Window gets destroyed**.
4. Configure the **Window gets destroyed** trigger.
 - a. Select **Start State** from the **Properties pane > Form Editor tab > Next state id drop-down list**.

This setting indicates that when the credentials are saved, the AccessProfile is not activated unless the application returns to its initial state, or unless the user logs out of the application.
 - b. Capture the Login window signature.

Click **Edit** below the **Signature of the window getting destroyed** field.

Drag the **Finder tool** and drop it to the Login window.

The generated signature is:

```
/child::wnd[@title="Login" and @class_name="#32770"]
```
 - c. Click **OK**.
5. Create a **Save credentials** action for the **Window gets destroyed** trigger.

Right-click on the **Window gets destroyed** trigger and select **Add action > Save credentials**.
6. Configure the **Save credentials** action.

Select the **Save credentials** action. The **default_capture_bag** is automatically selected from the **Properties pane > Form Editor tab > Account data bag id drop-down list**.
7. Create a **Window is activated** trigger for the state_looking_for_logon_result state.

Create the second trigger to enable auto-fill of information if user logon fails. If the logon fails, the **Window gets destroyed** trigger fails to fire.

Note: The state-engine cannot move forward as it continues to wait for a trigger to fire. Create another **Window is activated** trigger to fire when the logon prompt appears again.

 - a. Copy the **Window is activated** trigger from the **Start State**.

Select the **Window is activated** trigger that you created under the start state, right-click on the trigger node, and click **Copy**.
 - b. Paste the **Window is activated** trigger to the state_waiting_for_logon_result state.

Right-click `state_waiting_for_logon_result` state, and click **Paste**. The entire trigger (including actions) is copied.

8. Click the **Save selected data to file** icon from the **AccessStudio** toolbar to save the AccessProfile.
9. Select the **Upload selected data to IMS** icon from the **AccessStudio** toolbar to upload the AccessProfile to the IMS Server.
10. Click the **Start Test** icon from the **AccessStudio** toolbar to test the AccessProfile.
See “Testing AccessProfiles” on page 113 for detailed instructions on testing AccessProfiles.
It also contains a detailed description of the logs that are created for AccessProfiles when the application is running.

Example

The Patient Information Manager application is used as an example. See this description to confirm your understanding of the AccessProfile you created.

See the **Login** state engine as illustrated in the **States diagram** tab:

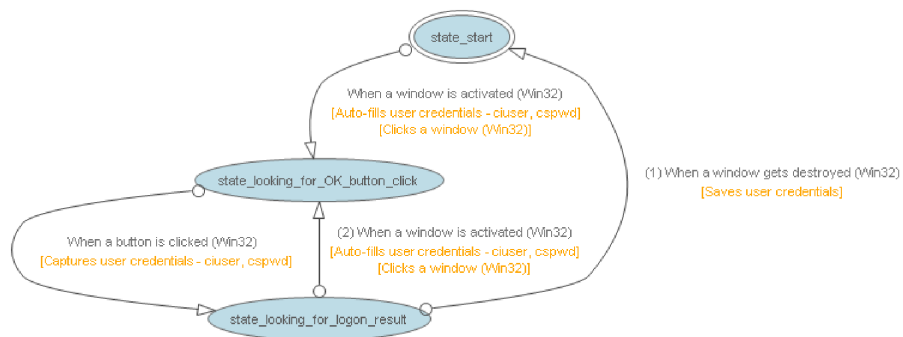


Figure 5. Sample state diagram of Patient Information Manager application.

Change password scenario

You can add a change password AccessProfile to your logon AccessProfile. Adding a change password AccessProfile enables automatic password change for the Change Password window of the sample application, Patient Information Manager.

About this task

The following diagram illustrates the states, triggers, and actions to create to enable automatic change password for this application.

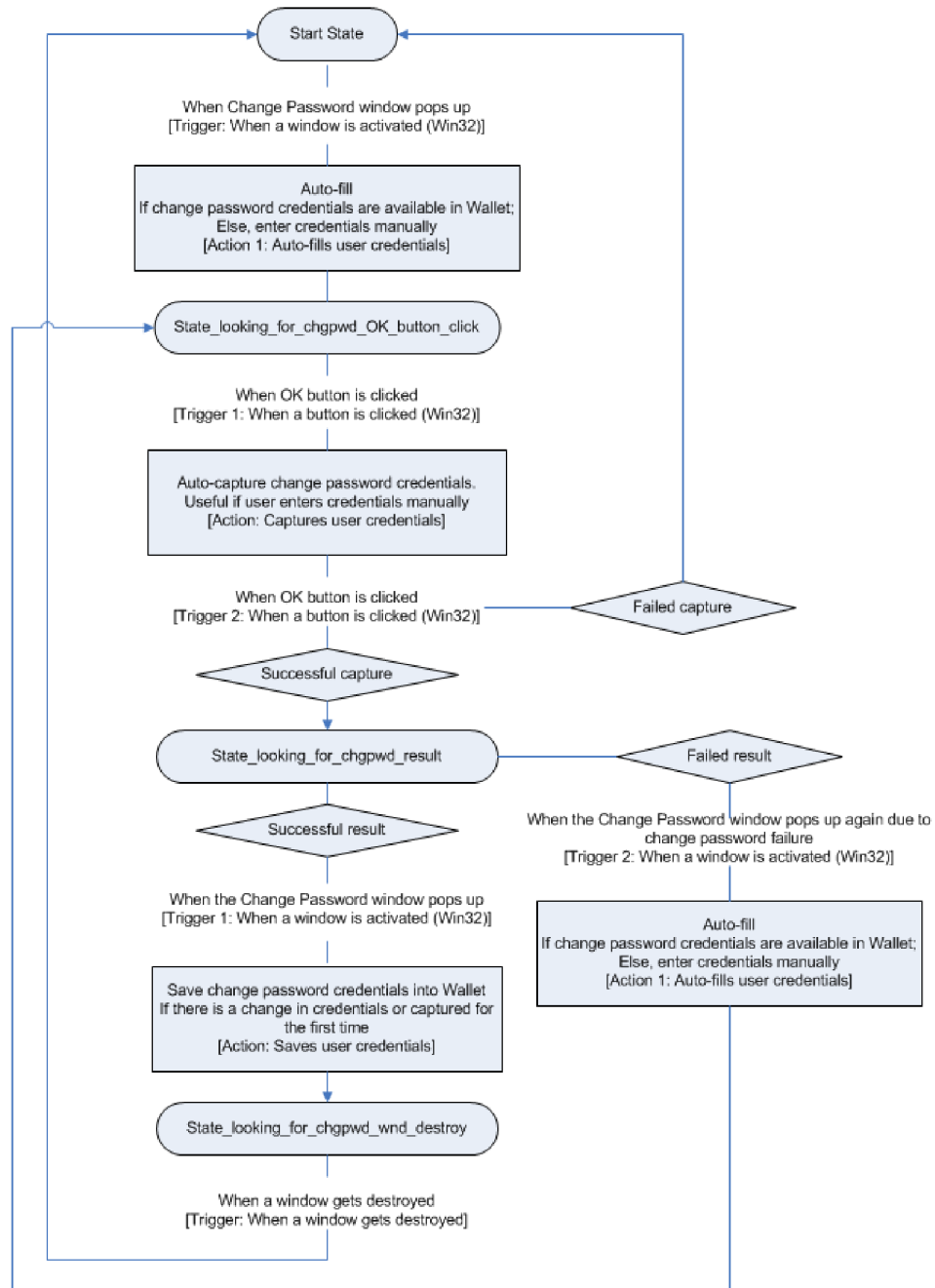


Figure 6. Sample workflow to enable automatic change password

Procedure

1. Launch AccessStudio.
2. Select **File > Open file** from the menu.
3. Browse for and open the logon AccessProfile you created, Logon_profile1.eas.
4. In the **States diagram** tab, create a state_looking_for_chgpwd_OK_button_click state.
 - a. From the **States diagram** toolbar, click the **New State** icon. This task generates a new state, **New State 1**.
 - b. Select the **New State 1** state.

- c. In the **Properties pane > Form Editor tab**, change the new state name into `state_looking_for_chgpwd_OK_button_click` in the **State Name** field.
5. Create a **Window is activated** trigger for the start state.

Right-click on start state and select **Add Trigger > Window is activated**. This trigger fires when the Change Password window is activated.

Specify the information for the trigger:

 - a. Select the **Window is activated** trigger.

In the **Properties pane > Form Editor tab**, select `state_looking_for_chgpwd_OK_button_click` as the state ID from the **Next state Id** list.
 - b. Capture identification information for the Change Password window of the Patient Information Manager application.
 - 1) Open the Patient Information Manager application.
 - 2) Go to **File > Change Password**.
 - c. In the **Properties pane Form Editor tab**, click **Edit** below the **Signature of the window getting the activate msg** field to display the Signature Window.

Drag the **Finder tool** and drop it to the **Change Password** window.

The generated signature is:

```
/child::wnd[@title="Change Password" and @class_name="#32770"]
```
 - d. Click **OK**.
 - e. Create an action to automatically insert the password in the relevant fields when the **Window is activated** trigger is fired.
 - 1) Right-click on the **Window is activated** trigger
 - 2) Select **Add Action > Inject credentials**.
6. Specify the information for **Inject credentials** action.
 - a. Select the **Inject credentials** action.
 - 1) In the **Properties pane > Form Editor tab**, specify the account data bag ID for the action.

An account data bag is a temporary virtual holder that stores the user account data (credentials) at the point of credential auto-fill, or capture.

Two IDs are provided by default.
 - 2) Select **default_injection_bag** from the **Account data bag id** list.
 - b. Capture the **Old Password** field signature.
 - 1) Expand **Capture fields** twistie.
 - 2) Click **Add** beside the **Capture fields** list.

This task expands the **Windows control** fields.
 - 3) Select **aditi_cspwd** from the **Account date item template id** list.
 - 4) Click **Edit** below the **Signature of the window for injection or capture** field.
 - 5) Drag it to the **Old Password** field of the Change Password window.
 - 6) Click **OK**.

The captured signature is:

```
/child::wnd[@title="Change Password" and @class_name="#32770"]/child::wnd[@class_name="Edit" and @ctrl_id=205]
```
 - c. Add an authentication service to the **Inject credentials** action.

- 1) In the **Properties pane > Form Editor tab**, expand the **Auth Info** twistie.
 - 2) In the **Add Auth Info** field, **Direct - Authentication Service** is selected by default. Click **Add**.
This task expands the **Direct - Authentication Service** field.
 - 3) Select the **authentication service ID** for the Patient Information Manager application.
7. Create the `state_looking_for_chgpwd_result` state.
 - a. From the **States diagram** toolbar, click the **New State** icon.
 - b. Select the **New State 1** state.
 - c. In the **Properties pane > Form Editor tab**, change the new state name into `state_looking_for_chgpwd_result` in the **State Name** field.
 8. Create a **Button is clicked** trigger for the `state_looking_for_chgpwd_OK_button_click` state.
 - a. Right-click on the `state_looking_for_chgpwd_OK_button_click` state.
 - b. Select **Add Trigger > Button is clicked**.
 9. Configure the **Button is clicked** trigger of the `state_looking_for_chgpwd_OK_button_click` state.
 - a. Select the **Button is clicked** action.
In the **Properties Pane > Form Editor tab**, select `state_looking_for_chgpwd_result` from the **Next state id** list.
 - b. Click **Edit** below the **Signature of the window receiving the command notification** field. This task opens the Signature window.
 - 1) Drag the **Finder tool** and drop it to the Change Password window.
 - 2) Click **OK**.
The generated signature is:
`/child::wnd[@title="Change Password" and @class_name="32770"]`
 - c. Click **Edit** below the **Signature of control** field.
 - 1) Drag the **Finder tool** and drop it to the **OK** button in the Change Password window.
 - 2) Click **OK**.
The generated signature is:
`/child::wnd[@title="Change Password" and @class_name="32770"]/child::wnd[@class_name="Button" and @ctrl_id=1]`
 - d. Create a **Capture credentials** action.
 - 1) Right-click on the **Button is clicked** trigger.
 - 2) Select **Add Action > Capture credentials**.
 10. Configure the **Capture credentials** action.
 - a. Select the **Capture credentials** action.
Expand **Capture fields** and click **Add** beside the **Capture fields** list. This task expands the Windows control fields.
 - b. Select `aditi_cspwd` from the **Account data item template id** list.
 - c. Click **Edit** below the **Signature of the window for injection or capture** field. This task opens the Signature window.
 - 1) Drag the **Finder tool** and drop it to the **New Password** field of the Change Password window.
 - 2) Click **OK**.

The generated signature is:

```
/child::wnd[@title="Change Password" and  
@class_name="32770"]/child::wnd[@class_name="Edit" and  
@ctrl_id=203]
```

- d. Click the **Auth Info** twistie to expand.

In the **Add Auth Info** field, **Direct - Authentication Service** is selected by default.

- 1) Click **Add**.
- 2) Select the **authentication service ID** for the Patient Information Manager application.
This task expands the **Direct - Authentication Service** field.
- 3) Select the **authentication service ID** for the Patient Information Manager application.

11. Create another **Button is clicked** trigger.

- a. Right-click on the `state_looking_for_chgpwd_OK_button_click` state, then select **Add trigger > Button is clicked**.
- b. Specify information for the **Button is clicked** trigger.
- c. Select the new **Button is clicked** trigger.

In the **Properties pane > Form Editor tab**, select `Start_State` as the state ID in the **Next State Id** field.

- d. Capture identification information for the Change Password window of the Patient Information Manager application.
 - 1) Click **Edit** below the **Signature of the window receiving the command notification** field to display the Signature window.
 - 2) Drag the **Finder tool** and drop it to the Change Password window.
 - 3) Click **OK**.

The generated signature is:

```
/child::wnd[@title="Change Password" and @class_name="32770"]
```

- e. Capture identification information for the **Cancel** button on the Change Password window.
 - 1) Click **Edit** below the **Signature of control** field to display the Signature window.
 - 2) Drag the **Finder tool** and drop it to the **Cancel** button in the Change Password window.
 - 3) Click **OK**.

The generated signature is:

```
/child::wnd[@title="Change Password" and  
@class_name="32770"]/child::wnd[@class_name="Button" and  
@ctrl_id=2]
```

12. Create a `state_looking_for_chgpwd_wnd_destroy` state.

- a. From the **States diagram** toolbar, click the **New State** icon.
- b. Select the **New State 1** state.
- c. In the **Properties pane > Form Editor tab**, change the new state name to `state_looking_for_chgpwd_wnd_destroy` in the **State Name** field.

13. Create a **Window is activated** trigger for the `state_looking_for_chgpwd_result` state.

- a. Right-click on the `state_looking_for_chgpwd_result` state.
- b. Select **Add trigger > Window is activated**.

14. Configure **Window is activated** trigger.

- a. Select **Window is activated**.

In the **Properties pane > Form Editor tab**, select `state_looking_for_chgpwd_wnd_destroy` from the **Next state id** list.

- b. Click **Edit** below the **Signature of the window getting the activate msg** field.

- 1) Drag the **Finder tool** and drop to the Success window.

The Success window displays after you have successfully changed the password.

- 2) Click **OK**.

The generated signature is:

```
/child::wnd[@title="Success" and @class_name="#32770"]
```

15. Create an action to save user credentials for the **Window is activated** trigger.

Right-click on the **Window is activated** trigger, select **Add action > Save credentials**.

16. Create another **Window is activated** trigger for the `state_looking_for_chgpwd_result` state.

- a. Right-click the `state_looking_for_chgpwd_result` state.

- b. Select **Add trigger > Window is activated**.

17. Configure the **Window is activated** trigger.

- a. Select the **Window is activated** trigger.

In the **Properties pane > Form Editor tab**, select `state_looking_for_chgpwd_OK_button_click` from the **Next state id** list.

- b. Click **Edit** below the **Signature of the window getting the active message** field.

- 1) Drag the **Finder tool** and drop it to the Change Password window.

- 2) Click **OK**.

The generated signature is:

```
/child::wnd[@title="Change Password" and @class_name="#32770"]
```

- c. Create an **Inject credentials** action.

Right-click **Window is activated** trigger and select **Add action > Inject credentials**.

18. Configure the **Inject credentials** action.

- a. Expand the **Inject fields** and click **Add** beside the **Inject fields** list.

- b. Select `aditi_cspwd` from the **Account data item template id** list.

- c. Click **Edit** below the **Signature of the window for injection or capture** field.

- d. Drag the **Finder tool** and drop it to the **Old Password** field of the Change Password window.

- e. Expand the **Auth Info** twistie. In the **Add Auth Info** field, **Direct - Authentication Service** is selected by default.

- 1) Click **Add**.

- 2) Select the **authentication service ID** for the Patient Information Manager application.

Note: You can copy and paste the **Inject credentials - cspwd** action of the start state since it contains similar data. Right-click **Inject credentials - cspwd** under the **Window is activated** trigger of the start state. Select **Copy**. Then, right-click the **Window is activated** trigger of the `state_looking_for_chgpwd_result` state. Select **Paste**.

19. Create the trigger for saving of user credentials (**Window gets destroyed** trigger) for the `state_looking_for_chgpwd_wnd_destroy` state.
 - a. Right-click on the `state_looking_for_chgpwd_wnd_destroy` state.
 - b. Select **Add Trigger > Advanced > Window gets destroyed**.
20. Configure the **Window gets destroyed** trigger.
 - a. Select **Start_State** from the **Next state id** list.
 - b. Click **Edit** below the **Signature of the window getting destroyed** field.
 Drag the **Finder tool** and drop it to the Change Password window.
21. Click the **Save selected data to file** icon from the AccessStudio toolbar to save the AccessProfile.
22. Select the **Upload selected data to IMS** icon from the AccessStudio toolbar to upload the AccessProfile to the IMS Server.
23. Click the **Start Test** icon from the AccessStudio toolbar to test the AccessProfile.
 See “Testing AccessProfiles” on page 113 for detailed instructions on testing AccessProfiles. It also contains detailed description of the logs that are created for AccessProfiles when the application is running.

Example

The Patient Information Manager application is used as an example. You can refer to this description to confirm your understanding of the AccessProfile you created.

See the **Change Password** state engine as illustrated in the **States diagram** tab.

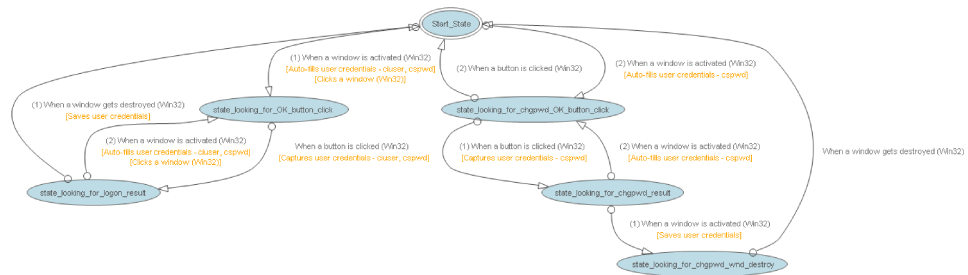


Figure 7. Sample state diagram of change password state engine

Checking and prompting for relogin

By default, an AccessAgent user who logs off from an application is prompted to log on again. The user clicks **OK**, then credentials are automatically injected.

About this task

Use the **Check and prompt for relogin** settings of AccessStudio to automatically inject the credentials of your user after application logoff.

Procedure

1. Open the AccessProfile of the application that you must modify.
2. In the **States** tab, right-click on the trigger which has the **Inject credential** action.
3. Select the **Inject credential** action.

4. In the **Properties** tab, go to **Advanced Options > Check and prompt for relogin**.
5. Select **Yes** from the **Check and prompt for relogin** list.

Note: Make sure that the **Override injection policy** is set to the default option, **Do not override**.

6. Click **Start Test**.
7. Run the application with the AccessProfile you modified. Log on, then logoff.

Note: If the credentials are automatically injected after the user logs off the application, the AccessProfile is set up correctly. If not, check the Message pane for AccessProfile error logs, return to the **Advanced Options** pane, and make the necessary corrections.

8. Click **Save**.

Adding fields for random passwords

IBM Security Access Manager for Enterprise Single Sign-On has a policy through which you can automatically fill random passwords during password change scenarios.

About this task

The random password fields feature in AccessStudio follows the same set of rules in the AccessAdmin.

Single sign-on takes effect only if the **Enable manual password change with random password (pid_auth_fortification_random_pwd_enabled)** policy is enabled on the user level in AccessAdmin.

Note: The Administrator can also include the **pid_auth_fortification_random_pwd_enabled** policy in a user policy template to apply the setting to all users or a group of users. See the policies in the *IBM Security Access Manager for Enterprise Single Sign-On Policies Definition Guide* for more information.

Procedure

1. Open an existing AccessProfile.
2. In the **States** tab, select the **Inject credentials** action from the **States diagram**.
3. Go to the **Properties pane > Form Editor tab** and click the **Random password fields** twistie to expand.
4. Select **Yes** from the **Generate new password** list.
5. Select **Windows control** from the **Random password fields** list.
6. Click **Add**.
7. Capture the new password signature of the application.
8. Select the appropriate option from the **Account data item template id** drop-down list.
9. Click **Edit** below the **Signature of the window for injection or capture** field. Drag the **Finder tool** and drop it on the **New Password** field.

Note: To prevent the user from overriding the auto-generated passwords, the Administrator can add a **Click a window** action. Capture the signature of the **OK** button in the change password AccessProfile, then save the action. The

next time the change password dialog box is displayed, the new password is automatically filled, and the **OK** button is automatically clicked.

Results

A random password field is automatically generated by the AccessProfile Generator if you profile a change password screen.

Using advanced AccessProfiles to meet custom requirements

IBM Security Access Manager for Enterprise Single Sign-On has a policy that allows you to automatically fill random passwords during password change scenarios.

For more information:

- “Customized triggers and actions using VBScript and JScript plug-ins”
- “Configuring support for existing applications using HLLAPI” on page 73

Customized triggers and actions using VBScript and JScript plug-ins

There are available built-in triggers and actions from AccessStudio. In most cases, the default sets of triggers and actions are sufficient.

AccessStudio enables customization through its VBScript and JScript plug-ins.

You can customize triggers and actions if you have unique requirements. You can use VBScript and JScript in AccessStudio to create custom triggers and actions. These plug-ins can interact with AccessAgent and target applications.

You can also use plug-in API in the scripts. See “Using plug-in API” on page 87.

For more information:

- “Customizing triggers”
- “Customizing actions” on page 72

Customizing triggers

You can customize triggers in AccessStudio if you do not find a specific condition in the predefined trigger options. Customized triggers can be created by adding different test conditions.

About this task

Use a condition to specify extra criteria that must be met before the trigger associated with it fires. The condition can be checking a property value against a string, a JScript, or a VBScript view. It meets the condition if the function returns true.

You can specify multiple conditions for the same trigger. In such a case, all conditions must be satisfied before the trigger fires.

Examples of test conditions are:

- Test property
- Test with script

Procedure

1. Select the trigger to customize.
2. In the **Properties pane > Form Editor tab**, click the **Conditions** twistie to expand.
3. Select a condition from the **Add Conditions** list.
4. Click **Add** to add a script. Use the **Check script return value** condition to specify a script to use for the test.

The condition is satisfied only if the script is evaluated as true.

Note: You can access all the local and global properties in the script.

5. Select a scripting language from the **Language** list. You can choose between VBScript and JScript.
6. Write the script in the Script panel, or click **Open Script Editor** to choose from a list of VBScript or JScript plug-ins.
See "Using plug-in API" on page 87 for details.

Example

The test condition must be specified in a function that is called test, which returns a Boolean value. Script to test if the property total is less than 10.

See the sample VBScript that runs a condition test. In this example, variable "test" is returned from the script.

```
Function Test

Set pc=runtime.GetPropertiesContainer()

b=pc.GetPropValue("total")

if b="" then
    test=True
else
    c=CInt(b)
    if c<10 then
        test=True
    else
        test=False
    end if
end if

end function
```

Customizing actions

Use AccessStudio to create customized actions if your action is not available in the ones that are provided in AccessStudio.

About this task

Customized actions can be created by writing scripts in VBScript or JScript. AccessStudio provides support for these scripts to interact with AccessAgent.

Procedure

1. In the **States** tab, right-click on the trigger and select **Add Action > Advanced > Run a VBScript or JScript**. The corresponding fields are displayed in the **Properties pane > Form Editor tab**.
2. Enter a name for the VBScript or JScript in the **Script Name** field. The **Run a VBScript or JScript** action name in the state engine is updated with the script name.
3. Enter the script code in the **Script** field.

The action is active when you upload the AccessProfile to the IMS Server.

Note: You can write scripts in VBScript or JScript. You can also click **Open Script Editor** to use plug-in API in the scripts. For more information, see "Using plug-in API" on page 87.

Example of a possible requirement and configuration for a customized action:

If a user enters the user name in any format, and you want to capture and save the user name in AccessAgent only in uppercase, create a customized action to ensure that the user name is always captured in the uppercase. A typical user credential capture would have only two actions: capture and save.

- a. To convert a captured user name into uppercase:
 - 1) Capture user name in account data bag.
 - 2) Transfer the captured user name into a property item.
 - 3) Convert the user name into uppercase by using the script.
 - 4) Transfer the converted user name back to the account data bag.
 - 5) Save.
- b. To do this task in AccessStudio:
 - 1) Create a **Button is clicked** trigger and specify the required information.
 - 2) Under this trigger, create a **Capture credentials** action.
In the **Properties pane > Form Editor tab**, select an ID from the **Account data bag id** list.
 - 3) Create a **Run a VBScript or JScript** action.
Change the language to JScript and enter the following code in the **Script** field:

```
var pc=runtime.GetPropertiesContainer();
var oldUser=pc.GetAccDataItem("default_capture_bag","aditi_ciuser",1);
pc.SetAccDataItem("default_capture_bag","aditi_ciuser",oldUser.toUpperCase(),1);
```
 - 4) Create a **Save credentials** action.
In the **Properties pane > Form Editor tab**, select the account data bag that you are using from the **Account data bag id** list.

Configuring support for existing applications using HLLAPI

High Level Language Application Programming Interface (HLLAPI) is a standard to access existing information and applications that are housed on IBM and Unisys mainframes, AS/400, UNIX/VMS, and others.

Examples of applications that provide HLLAPI are Attachmate EXTRA, Reflection, and IBM iSeries. Terminal Emulators are typically used to connect to these existing applications.

Note: The existing applications require configuration for HLLAPI to be activated. See the documentation of the application for more information.

For more information, see the following topics:

- “HLLAPI on terminal emulators”
- “Configuring HLLAPI support”

HLLAPI on terminal emulators

All installations of terminal emulators must have the following settings. Configuration for each application varies. See the documentation of the application for configuration information.

- **Session Short Name:** Unique name to identify a host session. This value must be a single letter from A to Z. T
- **Session Long Name:** Alternate name to identify a host session, up to eight characters.
- **HLLAPI DLL:** Dynamic Linked Library for HLLAPI functionality that is provided by the terminal emulator application.

Configuring HLLAPI support

You can enable automatic logon for mainframe and terminal applications that are HLLAPI-compatible by using AccessStudio. Automatic logon is supported for terminal emulators by using Enhanced Emulator HLLAPI.

Procedure

1. Configure the short name or long name of the host session.
Determine and configure the short name and long name of the host session on the existing application.
A short name is a unique identifier for the host session. The long name is an alternate identifier for the host session. See the documentation of the existing application for details on configuring the short and long names.
2. Determine the .DLL file of host application that provides HLLAPI support.
Determine the .DLL file that is provided by the application for 32-bit EHLLAPI-Enhanced support. See the documentation of the existing application for details.
3. Specify the application type and .DLL file information in AccessStudio.
 - a. In the **General Properties** tab, click to expand the **Extra Support Information** twistie.
 - b. Select the **Enable HLLAPI support for mainframe applications** check box.
 - c. Enter the relative path of the .DLL file in the **DLL relative path** field.
4. Configure session start information.
 - a. Right-click on the state and select **Add Trigger > Advanced > When a session starts**.

Note: You can also use the **Text is displayed (HLLAPI)** trigger for HLLAPI applications to see when a text is printed on the screen.

- b. Capture identification information for the application window:
 - 1) In the **Properties pane > Form Editor tab**, capture the signature information of the HLLAPI-enabled application window by clicking the **Finder tool** below the **Signature of the HLLAPI-enabled application window** field.

- 2) Drag the **Finder tool** and drop it on the field or window of the application you are configuring the AccessProfile.
The signature is automatically generated and displayed in the **Generated Signature** text box.
- c. Click the **Advanced Options** twistie to expand.
 - 1) Select queue actions from the **Queue actions** list.
 - 2) Specify the short name and long name information by using the corresponding fields provided. It is mandatory to specify the short name information.
5. Proceed to create states, triggers, and actions for auto-filling, auto-capturing, and saving credentials.

Signatures

Users can edit advanced AccessProfiles in AccessStudio. This feature allows users to edit the advanced AccessProfiles of an application by using signatures.

For more information about advanced AccessProfiles, see Chapter 4, “Advanced AccessProfiles,” on page 51.

For more information:

- “About signatures”
- “Supported axes” on page 76
- “Supported types” on page 76
- “Available operators” on page 77
- “Significance of the root node (‘/’)” on page 77
- “Executable signatures” on page 78
- “Signatures for windows” on page 79
- “Signatures for web pages” on page 80
- “Signatures for HTML” on page 82
- “Signatures for frames” on page 83
- “Signatures for Java windows” on page 85

About signatures

Signatures contain XML Path Language - a language that facilitates XML document navigation to select elements and attributes.

Signatures have a hierarchical structure or a tree representation of an XML document. You can navigate around the list items and select list items by various criteria. From this tree, you can access the elements, attributes, and list items of your .XML document.

Most elements that are recognized by AccessProfiles are part of a hierarchical structure. The signatures use the positioning of the element in the hierarchy, as well the properties of the element itself to correctly identify the element.

For example, you can specify an edit control in a login window by writing a signature, which not only refers to the properties of the control (for example, control ID), but also the property of its parent login window (for example, title).

Signatures in AccessProfiles can identify the following elements:

- **executable files**
Example: /child::exe[@exe_name="companypager.exe"]
Matches exes with name companypager.exe.
- **window elements (such as: edit control, buttons, and check box)**
Example: /child::wnd[@title="Login to CM"]/
child::wnd[@class_name#"*.BUTTON.*"]
Matches windows with title Login to CM and selects the descendant windows with class name that matches the regular expression *.BUTTON.* (# is for not case-sensitive match).
- **Web pages**
Example: /child::web[@domain="www.example.com" and @protocol=" http"]
Matches web pages from the URL with domain part equal to www.example.com and protocol equal to http.
- **HTML elements (such as: submit buttons, input controls, so on.)**
Example: /descendant::html[@tag_name="form" and @name=""]/
descendant::html[@tag_name="input" and @name="Passwd" and @type="password"]
The first HTML refers to the head or the body, after that a form descendant is found and then a descendant of that form of tag-name input and type password is searched.
- **Java window elements (such as: title, class name, window position, visibility status, size, so on.)**
Example: /child::jwnd[@title="Login" and @class_name="MyJFrame"]
Matches windows with title Login and class name MyJFrame.

These signatures can be edited in the **AccessProfile Generator** (for standard AccessProfiles), **General Properties** tab and **XML Editor** (for advanced AccessProfiles).

Supported axes

Use this table to view the supported axes that are used for advanced AccessProfiles.

Table 2. Supported axes

Axis name	Description
child::	Child of current control subset [Default axis]
parent::	Parent of current control subset
descendant::	Descendant of current control subset
ancestor::	Ancestor of current control subset
self::	Current control subset
rhs::	Right Side of control [first control]
lhs::	Left Side of control [first control]
top::	Top of control [first control]
btm::	Bottom of control [first control]

Supported types

Use this table to view the supported file types that are used for advanced AccessProfiles.

Table 3. Supported types

Name	Representation	Description
Executables	exe	To identify executable files.
Web pages	web	To identify web pages or Web sites.
Application (16-bit / Java /, and others)	task	To identify a 16-bit application, Java application, and other hosted applications
Windows	wnd	To identify windows.
Java window	jwnd	To identify windows of a Java application.
HTML element	html	To identify HTML elements inside web pages, including 'body' and 'head'.
HTML Form	form	To identify HTML forms.
HTML Input	input	To identify input fields in the HTML.
HTML Frame	frame	To identify the HTML frame that contains the document.
HTML document	document	To identify the document that contains the frameset or body and head.
HTML anchor	anchor	To identify the anchor elements that contain a name or ID (anchors without names or IDs are not included).
HTML image	Image	To identify all the images inside the HTML document.

Available operators

Use this table to view the operators that are used for advanced AccessProfiles.

Table 4. Available operators

Operator	Right side	Description
=	Numeric, String	Exact comparison. For strings, the comparison is not case-sensitive
~	String	Regex comparison, case-sensitive
#	String	Regex comparison, not case-sensitive
!=	Numeric	Not equal
!~	String	Not equal of regex comparison, case-sensitive
!#	String	Not equal of regex comparison, not case-sensitive
&	Numeric	Binary AND
!&	Numeric	Not equal of binary AND
	Numeric	Binary OR (NOT SUPPORTED IN THE CURRENT VERSION)
%%	Literal	The predefined literal between the two %s is translated to a numeric value
and	Logical	Logical and of two Boolean
or	Logical	Logical or of two Boolean

Significance of the root node ('/')

Use this table to view the type of root nodes that are used for advanced AccessProfiles.

Table 5. Root nodes

Type	Meaning
Executable	A hypothetical container of executable files in the system
Web pages	A hypothetical container of all open web pages in the system
HTML elements	A hypothetical container that contains all the HTML of a web page
Windows	A hypothetical container that contains all the top-level windows
HTML Form	Not allowed
HTML Input	Not allowed
HTML anchor	Not allowed
HTML image	Not allowed

Note: No attributes are available for the root element, so a signature cannot start with '/ self::'.

Executable signatures

Executable signatures identify executable files to determine whether an AccessProfile is loaded for it. Available attributes are **exe_name**, **company_name**, **file_version**, and others.

Table 6. Available attributes

Attribute Name	Description
exe_name	The name of the executable
internal_name	The internal name
language	The language supported*
original_file_name	The original file name
product_name	The product name
product_version	The product version
file_version	The file version
wnd_title	The windows title of the top-level window of application (16-bits / Java / Others)
class_name	The class name of the top-level window of application (16-bits / Java / Others)
task_name	The task name of the application (16-bits / Java / Others)

The .EXE attribute names reflect the version information as seen on explorer for any .EXE.

Examples:

```
/child::exe[@exe_name="companypager.exe"]
```

Matches exes with name companypager.exe.

```
/child::task[@exe_name="wowexec.exe" and @task_name="rumba.exe"]
```

Matches 16-bit application exes with name rumba.exe.

Signatures for windows

Signatures for windows identify window elements such as: edit control, buttons, and check boxes. A special form of windows is also identified by using this mechanism. The common attributes include title, class_name, ctrl_id, and others.

Unlike Web and .EXE signatures, Wnd signatures (and HTML signatures) support hierarchies. For example, you can identify parent, ancestor, descendant, and child relationships between windows.

Table 7. Available attributes

Attribute Name	Type	Description
window_style	Numeric	Windows styles. Typically used along with bit-wise operators (&,).
class_style	Numeric	Style of the class of window.
title	String	The title of the window as returned by the GetWindowText API. This string would be the title, as seen by the user, of the top-level window; the text of the button, and others. The value depends on the class of the window.
class_name	String	The class of the window.
ctrl_id	Numeric	Control ID of the child window with regard to its parent.
xpos	Numeric	Horizontal position of the window with regard to the parent window.
ypos	Numeric	Vertical position of the window with regard to the parent window.
is_foreground	Numeric (0/1)	is_foreground = 1 indicates that the window is a foreground window. A foreground window is the window that the user is working on.
is_visible	Numeric (0/1)	Visibility status of the window.
size	Numeric	The area of the window (WIDTH*HEIGHT). This attribute is useful on windows that cannot be resized.
window_ex_style	Numeric	Extended style of the window. This would be helpful in knowing which is extended style and which is not.
rel_xpos	Numeric	Considering the left-most position of all the windows, and numbering the windows from left to right starting from 1, the rel_xpos = n, represents the nth window. If two windows share the left x-pos, then they would get the same position, and the one afterward would have the next position. For example, for four windows with left-top pos (2,5), (4,10), (4,15) and (10,4). The position that is assigned is 1, 2, 2, 3.

Table 7. Available attributes (continued)

Attribute Name	Type	Description
rel_ypos	Numeric	Considering the top-most position of all the windows, and numbering the windows from top to bottom starting from 1, the rel_ypos = n, represents the nth window. If two windows share the left y-pos, then they would get the same position, and the one afterward would have the next position. For example, for four windows with left-top pos (2,5), (5,10), (4,10) and (10,15). The position that is assigned would be 1, 2, 2, 3.
file_path	String	Determine the file path of the executable file that belongs to the window.
file_name	String	Determine the file name of the executable file that belongs to the window.
control_name	String	Control name of the window control for .Net application.

Examples:

```
/child::wnd[@title="Login to CM" and @class_name="CMWindow"]/
descendant::wnd[@class_name#"*.BUTTON.*"]
```

The preceding example matches the windows with title **Login to CM** and selects the descendant windows with class name that matches the regular expression `*.BUTTON.*`. (# is for not case-sensitive match)

When the element pointed to by the signatures is used to set or get text, the `GetText` function on the window is used, so the behavior depends on the class of the window (element).

Signatures for web pages

Use `AccessProfile` signatures for web pages to determine whether an `AccessProfile` is loaded for it. Available attributes are domain, protocol, port, and others.

Available attributes

Attribute Name	Type	Description
domain	String	Domain part of the URL (before the slash character and after the protocol identifier)
protocol	String	Protocol used
query_string	String	Query string in the URL
path	String	Part of the URL after the domain and before the query string
port	Numeric	Port of connection
url	String	The complete URL

Examples:

/child::web[@domain="www.example.com" and @protocol="http"]

The preceding example matches Web pages from the URL with domain part equal to `www.example.com` and protocol equal to `HTTP`.

Signatures for Web: conceptual hierarchy

Signatures for Web represent the following organization. Use the typical parent, child, ancestor, and descendant axis to navigate from one node to another.

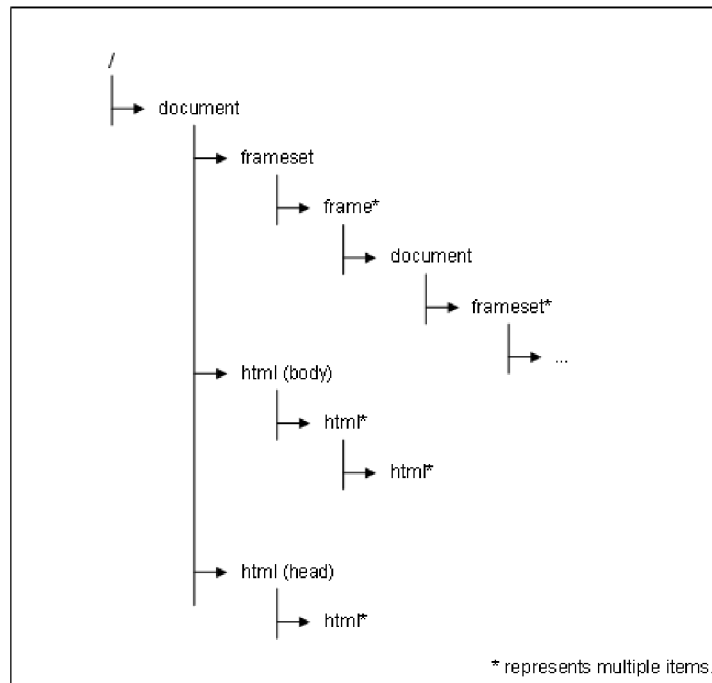


Figure 8. Web signature hierarchy

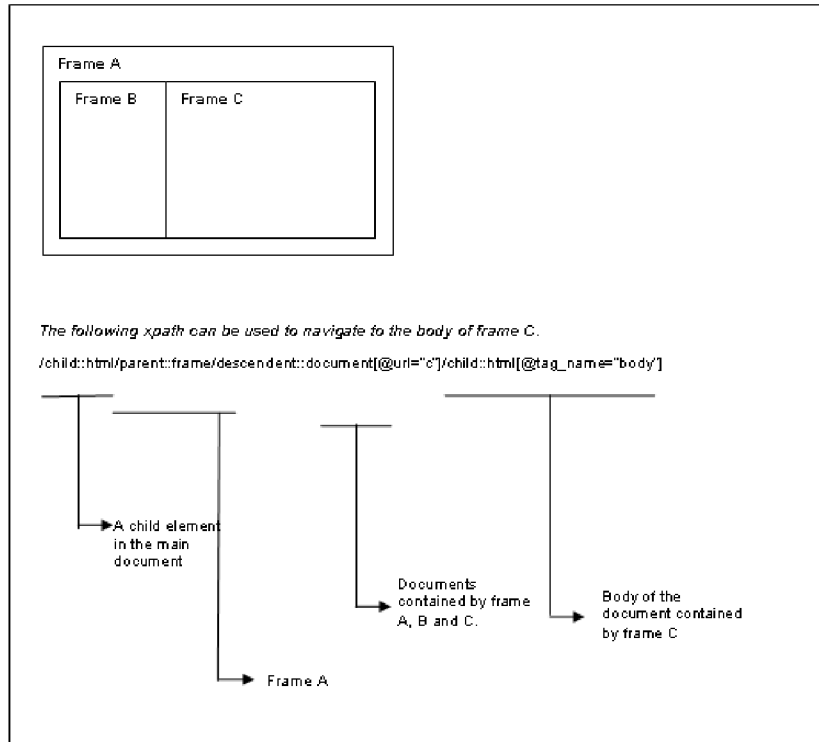


Figure 9. Sample of frameset navigation

Signatures for HTML

Signatures for HTML identify HTML elements in a web page, such as submit buttons, input controls, and others.

Note: There are also other miscellaneous attributes available for the element in the HTML DOM.

Value of the value attribute of an element is returned by using signatures to get the text of an HTML element.

Table 8. Available attributes (common elements)

Attribute Name	Description
tag_name	Identify the type of HTML element. For example, in the HTML snippet "" tag_name is "img".
inner_text	Inner text of any HTML element. For example, in the HTML snippet, "Login", Login is the inner text.
value	Value attribute of an HTML element. For example, in the HTML snippet, "<input value="user name here">", "user name here" is the value.
inner_html	Inner HTML of any HTML element. For example, in the HTML snippet "<p>login</p>" the inner html of 'p' element is login.

Examples [Common elements]:

```
/child::html/descendant::html[@tag_name="form"]/  
descendant::html[@tag_name="input" and @type="password"]
```

The first HTML refers to the head or the body. After that, a form descendant is found, and then a descendant of that form of tag-name input and type password is searched.

```
/child::html/descendant::html[@tag_name="form"]/  
child::html[@tag_name="table"]/descendant::html[@tag_name="b"] /@inner_text
```

Gets the inner text of a bold element that is inside a table that is within a form.

```
/child::html/descendant::html[@tag_name="form"]/  
child::html[@tag_name="table"]/descendant::html[@tag_name="b"]
```

Points to the element. When the text of this element is set the value attribute (if it exists), is set.

Signatures for frames

A frame displays multiple HTML documents in the same browser window. An iframe or inline frame, is an HTML document that is embedded inside another HTML document.

About frames

Each HTML document is called a frame. Each frame is independent of each other. The tags <frameset> and <frame> defines frames in a web page. An iframe HTML element is often used to insert content from another source, such as an advertisement, into a Web page. Use the tag <i frame> to embed another HTML document.

Frame and iframe are similar except that iframe is more flexible because you can embed another HTML document in the web page like embedding an image or any other HTML element. In a frame, you must define the structure of the framesets in a main web page. However, they are treated the same way in generating signatures.

Understand the following frame concepts before you generate web signatures for HTML elements inside another frame:

- Each frame in a web page has its own instance of state engine. These different state engines can be instantiated from the same AccessProfile or different AccessProfiles depending on the URL you specified in the site signature. However, you typically group all different frame URLs into one single AccessProfile.
- In your signature, root segment /child::html always refers to the document where the current state engine resides. If your state engine is loaded in an HTML page that resides in a frame, /child::html does not refer to the top-most document, but the HTML document that your state engine is in.
- /child::html/parent::frame/parent::frame points you to the top-most frame. Remember that you must use two parent or frame.
- To navigate to a particular document from the top-most frame, use the following signature:

```
/descendant::document[@url#...#]
```

One-level nested frame

For an ordinary one level nested frame, the *main.html* consists of two frame web pages, *page_a.html* and *page_b.html*

The following table provides the example signatures for a one-level nested frame:

Frame	Signature
<i>main.html</i>	<code><frameset cols="25%,75%"> <frame src="page_a.html"> <frame src="page_b.html"></frameset></code>
<i>page_a.html</i>	<code><html><body><form name="f_a"> <input name="i_a" type="text" value="Text here"></form></body></html></code>
<i>page_b.html</i>	<code><html><body><form name="f_b"> <input name="i_b" type="text" value="Text here"></form></body></html></code>

From anywhere in the page, use the following signature to navigate to the text box in *page_a.html*:

```
/child::html/parent::frame/parent::frame/descendent::document[@url#"*.page_a.*"]  
/descendent::html[@tag_name="form"]/descendent::html[@tag_name="input"]
```

Multi-nested frame

You might experience a complicated scenario where frame page contains another frameset, such as a multi-nested frame.

In the following example, the *main page* contains two frames, the first frame contains *page_a.html*, the second frame contains *page_b.html*. The frame *page_b.html* contains a frameset with two pages: *page_c.html* and *page_d.html*.

Frame	Signature
<i>main page</i>	<code><frameset cols="25%,75%"> <frame src="page_a.html"> <frame src="page_b.html"></frameset></code>
<i>page_a.html</i>	<code><html><body><form name="f_a"> <input name="i_a" type="text" value="Text here"></form></body></html></code>
<i>page_b.html</i>	<code><frameset rows="25%,75%"> <frame src="page_c.html"> <frame src="page_d.html"></frameset></code>
<i>page_c.html</i>	<code><html><body><form name="f_c"> <input name="i_c" type="text" value="Text here"></form></body></html></code>
<i>page_d.html</i>	<code><html><body><form name="f_d"> <input name="i_d" type="text" value="Text here"></form></body></html></code>

If you are sure that your state engine is loaded only in *page_c.html*, use the following signature to refer to an HTML element in *page_c.html*:

```
/descendent::html[@tag_name="input" and @name="i_c"]
```

If you are not sure about the frame your state engine is going to load, use the following signature:

```
/child::html/parent::frame/parent::frame/descendent::document[@url#"*.page_c.*"]  
/descendent::html[@tag_name="form"]/descendent::html[@tag_name="input"]
```

Note: The preceding signature is similar to the signature you use in the one-level nested signature. The `parent::frame` points to the top-level frame and irrespective of the depth you are in, you are going to navigate to the top-most frame.

Embedding another frame inside a web document using an iframe tag

The following table provides the example signatures. when you want to embed a *main.html* into another page *page_a.html*:

Frame	Signature
<i>main page</i>	<code><html><body><div>some text here</div> <iframe src="page_a.html" height="200"></iframe></body></html></code>
<i>page_a.html</i>	<code><html><body><form name="f_a"> <input name="i_a" type="text" value="Text here"></form></body></html></code>

In this scenario, two state engines loads, one in the top-level document, the other loaded by the *page_a.html*. If your state engine is at the top level on the document, use the following signature to reference to the input element:

```
/descendent::html[@tag_name="form" and @name="f_a"]/child::html[@tag_name="input" and @name="i_a" and @type="text"]
```

If your state engine is in the *page_a.html*, and if you want to reference to the HTML elements in *page_a.html*, do it as if there is no frame. But if you want to reference to an element in the top-level document, do the same thing as you did in frame. For example, from *page_a.html*, if we want to identify the *div* element, use the following signature:

```
/child::html/parent::frame/parent::frame/child::document[@url#"*.main.*"]  
/descendent::html[@tag_name="div"]
```

Cross-domain sites

A cross-domain web page is when an HTML document from one frame belongs to a different domain from the HTML document from another frame.

From Web script language (such as a JavaScript embedded inside an HTML source), you cannot access another cross-domain frame document for some security reasons. However, for the AccessStudio signature framework, cross-domain site signatures work because the signature evaluation algorithm uses a different approach that bypasses this security barrier. You do not have to worry about cross-domain sites when writing frame-related signatures.

Signatures for Java windows

You can use the following attributes such as the title, class name, window position, visibility status, and size to identify the Java window.

Available attributes

Attribute Name	Type	Description
title	String	The title of the Java window. We support to get the title for the following Java window class: java.awt.Frame, java.awt.Button, javax.swing.JFrame, javax.swing.JButton and all the other Java classes which supports getText() method. This support also includes all the extended classes of the mentioned classes.
class_name	String	The class name of the Java window.

Attribute Name	Type	Description
xpos	Numeric	Horizontal position of the window with regard to the parent window.
ypos	Numeric	Vertical position of the window with regard to the parent window.
is_visible	Numeric (0/1)	Visibility status of the window.
size	Numeric	The area of the window (WIDTH*HEIGHT). This attribute would be useful for windows that are not resizable.
rel_xpos	Numeric	<p>Considering the left-most position of all the windows, and numbering the windows from left to right starting from 1, the rel_xpos = n, represents the nth window.</p> <p>If two windows share the left x-pos, then they would get the same position, and the one afterward would have the next position.</p> <p>For example, for four windows with left-top pos (2,5), (4,10), (4,15) and (10,4). The position assigned would be 1, 2, 2, 3.</p>
rel_ypos	Numeric	<p>Considering the top-most position of all the windows, and numbering the windows from top to bottom starting from 1, the rel_ypos = n, represents the nth window.</p> <p>In case two windows share the left y-pos, then they would get the same position, and the one afterward would have the next position.</p> <p>For example, for four windows with left-top pos (2,5), (5,10), (4,10) and (10,15). The position assigned would be 1, 2, 2, 3.</p>

Example:

```
/child::jwnd[@title="Login" and @class_name="MyJFrame"]
```

This example matches windows with title "Login" and class name MyJFrame

Grouping of conditions

When using a parenthesis to group conditions inside a predicate, all the groups are inside their respective parentheses temporarily, including those groups that are originally excluded.

Example:

Wrong usage: `child::wnd[(@title="a" or @title="b") or @class_name="c"]` fails to evaluate because the class_name check is not grouped inside parenthesis.

Correct usage: `child::wnd[(@title="a" or @title="b") or (@class_name="c")]` evaluates as expected since the class name is grouped inside parenthesis.

Validating functions

Use AccessStudio to validate the accuracy or completeness of the functions you configure. These functions include AccessProfiles, Authentication Services, Applications, and Advanced Data functions like Account Data Templates. A red exclamation point displays beside a node that has an error in a function.

By default, the Messages pane displays at the bottom of AccessStudio application user interface. The pane displays the nature of the problem (a trigger or action) with an error.

To validate a function, select the trigger or action with a red exclamation point beside it to view the associated error description and settings in the Messages pane.

If a **Clicks a window** action has a red exclamation point beside it, an error occurred. Click that action, and the following details in the are displayed in the **General** tab:

The item has either incomplete or invalid content. Please check the Properties to make sure that all fields marked * are specified.
The following schema errors were detected.

1. The 'signature' element has an invalid value according to its data type.

The problem that is indicated in the Message pane also helps you to identify the solution. To resolve this problem, capture the signature of the corresponding data.

Using plug-in API

Customized triggers and actions can be created by writing plug-ins in VBScript or JScript. These plug-ins use the application programming interface (API) provided by AccessAgent.

See the following topics for more information:

- “Using plug-in API for customized triggers”
- “Using plug-in API for customized actions”
- “Plug-in API specifications” on page 88

Using plug-in API for customized triggers

Use AccessStudio to write plug-ins for customized triggers.

Procedure

1. See the Select the trigger to customize step to the Write the script in the Script Panel step of the “Customizing triggers” on page 71 procedure.
2. In the Script Editor, select from the list of plug-in API in the Type library information pane.
 - If you want to customize it further, you can also edit the script in the Script panel.
 - Click **Check Script for errors** to see whether you must edit your script.

Using plug-in API for customized actions

Use AccessStudio to write plug-ins for customized actions.

Procedure

1. See the In the States tab right-click on the trigger step to Enter the script code in the Script field step of the “Customizing actions” on page 72 procedure.
2. Then, see the In the Script Editor step of the “Using plug-in API for customized triggers” on page 87 procedure.

Plug-in API specifications

Use the following topics to view the plug-in API specifications for AccessStudio.

- “Runtime object”
- “Property Container class” on page 91
- “Property manager” on page 93
- “User data provider class” on page 95
- “Observer script debug object” on page 101
- “Window controller class” on page 101
- “ListControl” on page 108

Runtime object

Use these tables to view the plug-in API specifications for AccessStudio runtime objects.

runtime.GetPropertiesContainer()

Description	Gets the property container object which allows manipulation of property-items, as well as account-data-bags.
Details	Property items are like variables which can be used to store string values. Account data bags are also like variables which are used to store user credentials during the execution of the state machine.
Parameters	
Returns	The PropertyContainer object

runtime.CreateObject([in]BSTR bstrObjId)

Description	Creates a COM object with the specified ProgID or the CLSID. Note: This API can instantiate only the automation objects.
Details	This object is like the standard CreateObject API available in VBScript and JScript, except that it also supports taking in the CLSID of the class directly. This feature is useful while instantiating objects which do not have a ProgID defined.
Parameters	[IN] bstrObjId - The ProgID or the CLSID string for the object to be instantiated.
Returns	The automation interface of the object instantiated.

runtime.ShowMessageBox([in]BSTR bstrText, [in]BSTR bstrCaption)

Description	Shows a message box with no owner window with the provided text and caption. Note: The VBScript MsgBox and JScript alert function does not work in the plug-in.
Details	This API shows a standard Windows message box with an OK button without an owner window. The message box is not modal, which implies that the user can interact with the parent application even while the popup is present.

runtime.ShowMessageBox([in]BSTR bstrText, [in]BSTR bstrCaption)

Parameters	[IN] bstrText - The message to be displayed in the message box. [IN] bstrCaption - The title of the message box.
Returns	Nothing

runtime.GetHTMLDocument()

Description	Gets the current HTML Document object used by the state machine for the web page.
Details	You can use the HTML document to access/modify the HTML elements within the web page.
Parameters	
Returns	The automation interface of the HTML document.

runtime.GetUserDataProvider()

Description	Gets the User Data Provider object.
Details	The User Data Provider object can be used to get user and system policies, as well as to set user policies.
Parameters	
Returns	The UserDataProvider object

runtime.Sleep([in]long lTimeMil)

Description	Makes the thread wait for the given time.
Details	The calling thread sleeps for the time specified. Use the Wait for some time action instead of this API as that action just delays the execution of the next action in that trigger.
Parameters	[IN] lTimeMil - The time to wait in milliseconds.
Returns	Nothing

runtime.GetWindowController()

Description	Gets the window controller.
Details	The window controller object is used to get and set window properties.
Parameters	
Returns	The Window controller object

runtime.ShowModalMessageBox([in]long hParent, [in]BSTR bstrText, [in]BSTR bstrCaption)

Description	Shows a modal message box with the specified window as the owner. Note: The VBScript MsgBox and JScript alert function does not work in the plug-in.
Details	This API shows a standard Windows message box with an OK button. The message box becomes modal if the parent window handle is specified. This implies that the user is not able to interact with the parent application until this pop-up is closed. If the parent window handle is not specified, the window shown is modeless, like the ShowMessageBox function.

runtime.ShowModalMessageBox([in]long hParent, [in]BSTR bstrText, [in]BSTR bstrCaption)

Parameters	<p>[IN] hParent - The owner window handle. This handle cannot be a child window. If this value is 0, then the message box does not have an owner window.</p> <p>[IN] bstrText - The message to be displayed in the message box.</p> <p>[IN] bstrCaption - The title of the message box.</p>
Returns	Nothing

runtime.GetHTMLElementsFromXPath([in]IDispatch* pDoc, [in]BSTR bstrXPath)

Description	Evaluates the specified signature for the HTML document object and returns the array of HTML elements matched. Note: You can get the HTML document object by calling the GetHTMLDocument API of this object.
Details	
Parameters	<p>[IN] pDoc - The HTML Document2 object.</p> <p>[IN] bstrXPath - The signature string used for evaluation.</p>
Returns	The array of HTML Elements which matched the specified signature.

runtime.ReAuthPassCode([in, optional]long hParent)

Description	Pops up a reauthentication screen which asks the user to provide the password.
Details	
Parameters	[IN, OPTIONAL] hParent - The window to use as the owner window for the reauthentication pop-up. If this value is not specified or is 0, then the pop-up menu has no owner window.
Returns	VARIANT_FALSE (-1) if the reauthentication succeeds, VARIANT_TRUE (0) if the reauthentication fails.

runtime.GetBrowserObjectFromHTMLDocument([in]IDispatch* pDoc)

Description	Gets the WebBrowser object for a given HTML document object. Note: You can get the HTML Document by using the GetHTMLDocument API.
Details	The web browser object gives facilities to control the browser window containing the specified HTML document.
Parameters	[IN] pDoc - The HTML Document object.
Returns	The WebBrowser object which implements the IWebBrowser2 interface

runtime.GetAccessProfileId()

Description	Gets the current ID of the AccessProfile.
Details	
Parameters	
Returns	The AccessProfile ID string

SetTextOutMonitoringState([in] VARIANT_BOOL bEnabled)

Description	Do not enable or enable text-monitoring for an application assuming that the profile has mainframe support enabled.
Details	This task is done in rare cases when the text-out monitoring has a performance or behavior side-effect on the application, in which case it makes sense not to enable it after the SSO is performed, assuming no more screen text-outs must be monitored.
Parameters	[IN] bEnabled - 1 for enabling, 0 for not enabling.
Returns	Nothing

GetTextOutMonitoringState([out,retval] VARIANT_BOOL *pbEnabled)

Description	Checks whether screen text monitoring is enabled for an application.
Details	
Parameters	
Returns	1 for enabled, 0 for not enabled.

GetObsScriptDebugObject([out,retval] IObsScriptDbgSupport pRet)**

Description	Gets the ObsScriptDebug object used to write log messages for debugging purposes.
Details	
Parameters	
Returns	

Property Container class

Use these tables to view the plug-in API specifications for the AccessStudio Property Container class.

GetPropValue([in]BSTR bstrPropName, [in, optional]VARIANT varUseLocalBag)

Description	For a given property item, returns its value. Note: You can specify whether to use the local or global property bag. The local property bag holds property items for that state machine only, while the system property bag stores items on a system level basis.
Details	
Parameters	[IN] bstrPropName - The property item name to fetch value. [IN, OPTIONAL] varUseLocalBag - If specified as VARIANT_TRUE (-1) or if not specified then use the local bag. If specified as 0, use the global property bag.
Returns	The string value of the property item. If no property item is found, it returns an empty string.

SetPropValue([in]BSTR bstrPropName, [in]BSTR bstrVal, [in, optional]VARIANT varUseLocalBag)

Description	Sets the value of a given property item. Note: You can specify whether to use the local or global property bag. The local property bag holds property items for that state machine only, while the system property bag stores items on a system level basis.
Details	

SetPropValue([in]BSTR bstrPropName, [in]BSTR bstrVal, [in, optional]VARIANT varUseLocalBag)

Parameters	<p>[IN] bstrPropName - The property item name to fetch value.</p> <p>[IN] bstrVal - The string value to set the property item.</p> <p>[IN, OPTIONAL] varUseLocalBag - If specified as VARIANT_TRUE (-1) or if not specified, then use the local bag. If specified as 0, use the global property bag.</p>
Returns	Nothing

GetAccDataItem([in]BSTR bstrAccDataBagID, [in]BSTR bstrItemID, [in, optional]VARIANT varUseLocalBag)

Description	<p>Gets an account-data bag item from a specified account-data bag.</p> <p>Note: This API can throw an exception if it is not able to decrypt the password field properly.</p>
Details	<p>You can specify whether to use the local or global account-data bag. The local account-data-bag holds account-data items for that state-machine only, while the system account-data bag stores items on a system level basis. This API returns the decrypted password if the password item is specified for retrieval.</p>
Parameters	<p>[IN] bstrAccDataBagID - The account-data-bag.</p> <p>[IN] bstrItemID - The account-data-itemtemplate-id. This parameter is case-sensitive.</p> <p>[IN,OPTIONAL] varUseLocalBag - If specified as VARIANT_TRUE (-1) or if not specified, then use the local bag. If specified as 0, use the global property bag.</p>
Returns	The account-data-bag-item value string.

SetAccDataItem([in]BSTR bstrAccDataBagID, [in]BSTR bstrItemID, [in]BSTR bstr- Val, [in, optional]VARIANT varUseLocalBag)

Description	<p>Sets an account-data bag item for a given account-data bag.</p> <p>Note: This API can throw an exception if the account-data-item-template-id specified does not belong to this bag.</p>
Details	<p>You can specify whether to use the local or global account-data bag. The local account-data-bag holds account-data items for that state-machine only, while the system account-data bag stores items on a system level basis. This API returns the decrypted password if the password item is specified for retrieval.</p>
Parameters	<p>[IN] bstrAccDataBagID - The account-data-bag.</p> <p>[IN] bstrItemID - The account-data-itemtemplate-id. This parameter is case-sensitive.</p> <p>[IN,OPTIONAL] varUseLocalBag - If specified as VARIANT_TRUE (-1) or if not specified, then use the local bag. If specified as 0, use the global property bag</p>
Returns	Nothing

TransferAccDataBag([in]BSTR bstrSourceBagID, [in]BSTR bstrTargetBagID, [in, optional]VARIANT varSourceUseLocalBag, [in, optional]VARIANT varTargetUseLocalBag)

Description	Copies the contents of an account-data bag to another account-data-bag. Note: If the target bag is already filled, its contents are replaced.
Details	
Parameters	[IN] bstrSourceBagID - The source bag ID. [IN] bstrTargetBagID - The target bag ID. [IN,OPTIONAL] varSourceUseLocalBag - If specified as VARIANT_TRUE (-1) or if not specified, then use the local bag. If specified as 0, use the global property bag.
Returns	Nothing

IsAccDataBagExist([in]BSTR bstrAccDataBagID, [in, optional]VARIANT varUseLocalBag)

Description	Checks if the account data bag exists.
Details	
Parameters	[IN] bstrAccData- BagID - The account-data bag ID to test for existence. [IN,OPTIONAL] - If specified as VARIANT_TRUE (-1) or if not specified, then use the local bag. If specified as 0, use the global property bag.
Returns	VARIANT_TRUE (-1) if accountdata bag exists, otherwise returns VARIANT_FALSE (0)

IsAccDataBagItemExist([in]BSTR bstrAccDataBagID, [in]BSTR bstrItemID, [in, optional]VARIANT varUseLocalBag)

Description	Checks if an account-data-bag-item inside an account data bag exists.
Details	
Parameters	[IN] bstrAccDataBagID - The account-data bag ID. [IN] bstrItemID - The account-data-item ID to test for existence. [IN,OPTIONAL] - If specified as VARIANT_TRUE (-1) or if not specified then use the local bag. If specified as 0, use the global property bag.
Returns	VARIANT_TRUE (-1) if account-data-bag exists, otherwise returns VARIANT_FALSE (0)

Property manager

Use these tables to view the plug-in API specifications for AccessStudio property manager.

SetAccDataBagAuthId([in] BSTR bstrAccDataBagID, [in] BSTR bstrAuthId, [in,optional] VARIANT varUseLocalBag, [out,retval] VARIANT_BOOL* pbRet)

Description	Set the authentication-service ID inside an account-data-bag
Details	

SetAccDataBagAuthId([in] BSTR bstrAccDataBagID, [in] BSTR bstrAuthId, [in,optional] VARIANT varUseLocalBag, [out,retval] VARIANT_BOOL* pbRet)

Parameters	<p>[IN] bstrAccDataBagID - The account-data-bag ID to set authentication service ID.</p> <p>[IN] bstrAuthId - The auth-ID to set. Note: The authservice for this authentication service ID is present.</p> <p>[IN,OPTIONAL] varUseLocalBag - Whether the bag specified is a local bag, default is yes.</p>
Returns	<p>Success = 1</p> <p>Failure = 0</p>

GetAccDataBagAuthId([in] BSTR bstrAccDataBagID, [in,optional] VARIANT varUseLocalBag, [out,retval] BSTR* bstrAuthId)

Description	Get the authentication-service- ID from a bag
Details	
Parameters	<p>[IN] bstrAccDataBagID - The account-data-bag ID to fetch the authentication service ID.</p> <p>[IN,OPTIONAL] varUseLocalBag - Whether the bag specified is a local bag, default is yes.</p>
Returns	The authentication service ID if present, otherwise an empty string is returned.

SetAccDataBagADT([in] BSTR bstrAccDataBagID, [in] BSTR bstrADTID, [in,optional] VARIANT varUseLocalBag, [out,retval] VARIANT_BOOL* pbRet)

Description	Set the account data template ID for a bag
Details	
Parameters	<p>[IN] bstrAccDataBagID- The account-data-bag-id to set the account data template ID.</p> <p>[IN] bstrADTID - The account data template ID to set, for example 'adti_ciuser_cspwd'.</p> <p>[IN,OPTIONAL] varUseLocalBag - Whether the bag specified is a local bag, default is yes.</p>
Returns	<p>Success = 1</p> <p>Failure = 0</p>

GetAccDataBagADT([in] BSTR bstrAccDataBagID, [in,optional] VARIANT varUseLocalBag,[out,retval] BSTR* bstrADTID)

Description	Get account data template ID from an account data bag
Details	
Parameters	<p>[IN] bstrAccDataBagID - The account-data-bag ID to fetch the account data template ID.</p> <p>[IN,OPTIONAL] varUseLocalBag - Whether the bag specified is a local bag, default is yes.</p>
Returns	The account data template ID

CreateAccountDataBag([in] BSTR bstrAccDataBagID, [out, retval] int * iRet);

Description	Create an account-data-bag in the memory of the process. You can only create a local bag.
Details	
Parameters	[IN] bstrAccDataBagID - The bag ID to create
Returns	0 for error, 1 for ok, 2 for already exists

GetAccountData([in] BSTR bstrTragetBagID, [in] BSTR bstrFilterBagID, [out, retval] int * iRet)

Description	Take the contents of the filter bag to fetch matching data from the Wallet. The filter bag can be incomplete (for example, if the filter bag only contains an auth-service and a third-field), then all the credentials matching these two criteria are fetched. Note: Only the first match is fetched.
Details	
Parameters	[IN] bstrTargetBagID - The target bag to populate with the search result. [IN] bstrFilterBagID - The bag containing the search criteria.
Returns	0 for nothing found, negative values for other errors, a positive value indicates the number of actual matches found.

[id(14), helpstring("Get a random matched account data")] GetRandomAccountData([in] BSTR bstrTragetBagID, [in] BSTR bstrFilterBagID, [out, retval] int * iRet)

Description	Take the contents of the filter bag to fetch matching data from the Wallet. If more than one account-data is found, then select one of the found account-data at random.
Details	This API is used in a specialized case where multiple accounts are shared between some people. Since all accounts have the same privilege, it is possible to inject any account present.
Parameters	[IN] bstrTargetBagID - The target bag to populate with the search result. [IN] bstrFilterBagID - The bag containing the search criteria.
Returns	0 for nothing found, negative values for other errors, a positive value indicates the number of actual matches found.

User data provider class

Use these tables to view the plug-in API specifications for AccessStudio user data provider class.

GetGenericPolicy([in]BSTR bstrPolicyID)

Description	<p>Gets the resolved string array value of a custom policy. Note: For string return values, the string is pushed as the first item of the string array.</p> <p>For integer values, the integer is cast to a string, and then pushed as the first item in the array.</p> <p>For array of integers, each integer is cast individually and pushed to the array.</p> <p>For Boolean values, 0 corresponds to the value false, and a 1 corresponds to the value true as the first item in the array.</p>
Details	
Parameters	[IN] bstrPolicyID - The policy ID to return a value. This parameter is case-sensitive.
Returns	A string array of the policy value. If the policy type is not a string array, it would return an empty array.

GetAuthPolicy([in]BSTR bstrAuthID, [in]BSTR bstrPolicyID)

Description	<p>Gets the resolved value of an authentication service-specific policy. Note: For string return values, the string is pushed as the first item of the string array.</p> <p>For integer values, the integer is cast to a string, and then pushed as the first item in the array.</p> <p>For array of integers, each integer is cast individually and pushed to the array.</p> <p>For Boolean values, 0 corresponds to the value false, and a 1 corresponds to the value true as the first item in the array.</p>
Details	
Parameters	[IN] bstrAuthID - The authentication-service ID. This parameter is case-sensitive. [IN] bstrPolicyID - The policy ID to return a value. This parameter is case-sensitive.
Returns	A string array of the policy value. If the policy type is not a string array, it would return an empty array.

GetAppPolicy([in]BSTR bstrAppID, [in]BSTR bstrPolicyID)

Description	<p>Gets the resolved value of an application-specific policy. Note: For string return values, the string is pushed as the first item of the string array.</p> <p>For integer values, the integer is cast to a string, and then pushed as the first item in the array.</p> <p>For array of integers, each integer is cast individually and pushed to the array.</p> <p>For Boolean values, 0 corresponds to a false, and a 1 corresponds to the value true as the first item in the array.</p>
--------------------	--

GetAppPolicy([in]BSTR bstrAppID, [in]BSTR bstrPolicyID)

Details	
Parameters	<p>[IN] bstrAppID: The application ID. This parameter is case-sensitive.</p> <p>[IN] bstrPolicyID: The policy ID to return a value. This parameter is case-sensitive.</p>
Returns	A string array of the policy value. If the policy type is not a string array, it would return an empty array.

GetAuthAppPolicy([in]BSTR bstrAuthID, [in]BSTR bstrAppID, [in]BSTR bstrPolicyID)

Description	<p>Gets the resolved value of an authentication-service and application policy. Note: For string return values, the string is pushed as the first item of the string array.</p> <p>For integer values, the integer is cast to a string, and then pushed as the first item in the array.</p> <p>For array of integers, each integer is cast individually and pushed to the array.</p> <p>For Boolean values, 0 corresponds to a false, and a 1 corresponds to the value true as the first item in the array.</p>
Details	
Parameters	<p>[IN] bstrAuthID - The authentication-service ID. This parameter is case-sensitive.</p> <p>[IN] bstrAppID - The application ID. This parameter is case-sensitive.</p> <p>[IN] bstrPolicyID - The policy ID to return a value. This parameter is case-sensitive.</p>
Returns	A string array of the policy value. If the policy type is not a string array, it would return an empty array.

SetUserGenericPolicy([in]BSTR bstrPolicyID, [in]VARIANT parrayPolicyVal)

Description	<p>Sets a custom policy in the user scope. The value set is as string array. Note: For setting a string, push the string as the first item of the string array.</p> <p>For setting an integer, cast the integer as a string, and then push it as the first item in the array.</p> <p>For setting an array of integers, cast them to strings individually and push them to the array.</p> <p>For Boolean values, type 0 for false and 1 for true as the first item in the array.</p>
Details	
Parameters	<p>[IN] bstrPolicyID - The policy ID.</p> <p>[IN] parrayPolicyVal - An array of string values to set.</p>
Returns	Nothing

SetUserAuthPolicy([in]BSTR bstrAuthID, [in]BSTR bstrPolicyID, [in]VARIANT parrayPolicyVal)

Description	<p>Sets an authentication-service policy in the user-scope. The policy value must be a string array.</p> <p>Note: For setting a string, push the string as the first item of the string array.</p> <p>For setting an integer, cast the integer as a string, and then push it as the first item in the array.</p> <p>For setting an array of integers, cast them to strings individually and push them to the array.</p> <p>For Boolean values, push 0 for false and 1 for true as the first item in the array.</p>
Details	
Parameters	<p>[IN] bstrAuthID - The authentication-service ID. This parameter is case-sensitive.</p> <p>[IN] bstrPolicyID - The policy ID. This parameter is case-sensitive.</p> <p>[IN] parrayPolicyVal - An array of string values to set.</p>
Returns	Nothing

SetUserAppPolicy([in]BSTR bstrAppID, [in]BSTR bstrPolicyID, [in]VARIANT parrayPolicyVal)

Description	<p>Sets an application policy in the user-scope. The policy value must be a string array.</p> <p>Note: For setting a string, push the string as the first item of the string array.</p> <p>For setting an integer, cast the integer as a string, and then push it as the first item in the array.</p> <p>For setting an array of integers, cast them to strings individually and push them to the array.</p> <p>For Boolean values, push 0 for false and 1 for true as the first item in the array.</p>
Details	
Parameters	<p>[IN] bstrAppID - The application ID. This parameter is case-sensitive.</p> <p>[IN] bstrPolicyID - The policy ID. This parameter is case-sensitive.</p> <p>[IN] parrayPolicyVal - An array of string values to set.</p>
Returns	Nothing

SetUserAuthAppPolicy([in]BSTR bstrAuthID, [in]BSTR bstrAppID, [in]BSTR bstrPolicyID, [in]VARIANT parrayPolicyVal)

Description	<p>Sets an authentication-service and application policy in the user-scope. The policy value must be a string array.</p> <p>Note: For setting a string, push the string as the first item of the string array.</p> <p>For setting an integer, cast the integer as a string, and then push it as the first item in the array.</p> <p>For setting an array of integers, cast them to strings individually and push them to the array.</p> <p>For Boolean values, push 0 for false and 1 for true as the first item in the array.</p>
Details	
Parameters	<p>[IN] bstrAuthID - The authentication-service ID. This parameter is case-sensitive.</p> <p>[IN] bstrAppID - The application ID. This parameter is case-sensitive.</p> <p>[IN] bstrPolicyID - The policy ID. This parameter is case-sensitive.</p> <p>[IN] parrayPolicyVal - An array of string values to set.</p>
Returns	Nothing

ResolveAuthIndirect([in] VARIANT oArrayGroups, [in] BSTR bstrMatchString, [in] BSTR bstrAccountDataTemplateId, [out,retval] BSTR* bstrAuthId)

Description	<p>Resolves an authentication service ID indirectly by matching a given text with the server-urls of authentication services currently loaded.</p> <p>Note: This API is like the standard indirect-auth-info available under the capture or inject actions, except that instead of pointing to a UI element to pick the string, you can pass any arbitrary string so search against the server-urls.</p>
Details	If an authentication service is not found, a new authentication service is created for the user with the match-string sent as the ID.
Parameters	<p>[IN] oArrayGroups - A VB Array of authentication service group IDs under which the search happens. If this parameter is null, then all the currently loaded authentication services are searched.</p> <p>[IN] bstrMatchString - The string to match the authentication- service server-urls.</p> <p>[IN] bstrAccountDataTemplateId - The account-data-template ID for the authentication-service. This parameter is only used if no authentication-service was found and the API created an authentication service.</p>
Returns	The ID of the found or newly created authentication service.

GetAuthTemplateID([in] BSTR bstrAuthId, [out,retval] BSTR* bstrADTI)

Description	Gets the account data template ID corresponding to the authentication service ID passed.
Details	
Parameters	[IN] bstrAuthId - The authentication service ID for which the account-data-template- ID is to be retrieved.

GetAuthTemplateID([in] BSTR bstrAuthId, [out,retval] BSTR* bstrADTI)

Returns	The account-data-template-ID of the authentication-service-ID passed. If the authentication-service-ID is not found, the API returns an empty string.
----------------	---

SaveAccountData([in] BSTR bstrAuthId, [in] VARIANT oItemIDArray, [in] VARIANT oValArray, [out,retval] AccountDataProviderRetcodes *pnRet)

Description	Saves account data corresponding to a given authentication service ID.
Details	
Parameters	<p>[IN] bstrAuthId - The authentication service ID for which the account-data is to be saved.</p> <p>[IN] oItemIDArray - The array of account-data-itemtemplate-ids which have corresponding data to be stored in the Wallet. The order of this ID array and the Value array (see next parameters) match.</p> <p>For example, if the first item in the [IN] oValArray is the user name, then the first item in this array is 'aditi_ciuser', and so on.</p> <p>[IN] oValArray- The array of values to be stored for the corresponding IDs in the [IN] oItemIDArray.</p>
Returns	<p>An integer, with the following values:</p> <p>Success =1</p> <p>Invalid_ADT = -1</p> <p>Invalid_ADT_Item = -2</p> <p>Fail = -3</p>

SetAppAuthLink([in] BSTR bstrAuthId,[in] BSTR bstrAppId,[out,retval] VARIANT_BOOL* pbRet);

Description	Creates an application-authentication-service link object for this user.
Details	This API is primarily used during migration from a third-party SSO software, where we want to transfer over not only the credentials, but also details regarding which application was used to create which authentication service.
Parameters	<p>[IN] bstrAuthId - The authentication service ID</p> <p>[IN] bstrAppId - The application ID</p>
Returns	<p>Success =1</p> <p>Fail = 0</p>

DeleteCredentialFromWallet([in] BSTR bstrAuthId,[in] VARIANT oItemIDArray,[in] VARIANT oValArray,[out,retval] AccountDataProviderRetcodes *pnRet)

Description	Deletes the credential corresponding to a given authenticator and key items from the Wallet.
Details	
Parameters	<p>[IN] bstrAuthId - The authentication service ID.</p> <p>[IN] oItemIDArray - An array containing the account-data-item-template ID corresponding to the values in the [IN] oItemValArray parameter.</p> <p>[IN] oItemValArray - An array containing the values of the account-data-item-templates specified in the [IN] oItemIDArray parameter.</p>

DeleteCredentialFromWallet([in] BSTR bstrAuthId,[in] VARIANT oltemIDArray,[in] VARIANT oValArray,[out,retval] AccountDataProviderRetcodes *pnRet)

Returns	1 for success 0 or a lesser value for error
----------------	--

Observer script debug object

Use these tables to view the plug-in API specifications for AccessStudio observer script debug objects.

WriteRealTimeLog([in] BSTR strMessage)

Description	Writes a log line in the real-time logs tab of AccessStudio for this process.
Details	
Parameters	[IN] strMessage - The log line to write.
Returns	Nothing

WriteDebugViewString([in] BSTR strMessage)

Description	Writes a log line to the debug output stream. This object can be viewed by using tools such as DebugView.
Details	
Parameters	[IN] strMessage - The log line to write.
Returns	Nothing

WriteObserverLog([in] BSTR strMessage,[in] int nLevel)

Description	Writes a log line in the Observer logs written in the standard AccessAgent logs directory. The name of the log file is aa_observer.log.
Details	
Parameters	[IN] strMessage - The log line to write. [IN] nLevel - The log-level to use to write the log. 1 corresponds to low, 2 to medium and 3 to high.
Returns	Nothing

Window controller class

Use these tables to view the plug-in API specifications for AccessStudio Window controller class.

GetHWNDFromXPath([in]BSTR bstrWndXPath)

Description	For a given signature string, evaluates the matching window and return it.
Details	
Parameters	[IN] bstrWndXPath - The signature of the window to evaluate.
Returns	An unsigned long representing the window found, 0 if window is not found.

GetWindowText([in]ULONG hWnd)

Description	For a given handle to the window, returns its text. Note: The maximum length of text returned is 500 characters. This function only works in the same process as the caller process.
--------------------	--

GetWindowText([in]ULONG hWnd)

Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	An empty string in case the window does not have any text, or if the window is not found or in a different process.

GetWindowRect([in]ULONG hWnd, [out]long* ITLX, [out]long* ITLY, [out]long* IBRX, [out]long* IBRY)

Description	Returns the screen-coordinates of the window specified through its window handle. Note: This API throws an exception in case the window handle is not correct.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long. [OUT] ITLX - The x-coordinate of the top left corner. [OUT] ITLY - The y-coordinate of the top left corner. [OUT] IBRX - The x-coordinate of the bottom right corner. [OUT] IBRY - The y-coordinate of the bottom right corner.
Returns	Nothing

GetWindowClientRect([in]ULONG hWnd, [out]long* ITLX, [out]long* ITLY, [out]long* IBRX, [out]long* IBRY)

Description	Returns the client-coordinates of the window specified through its window handle. Note: This API throws an exception in case the window handle is not correct.
Details	
Parameters	[IN] hWnd : The handle to the window as unsigned long. [OUT] ITLX - The x-coordinate of the top left corner. [OUT] ITLY - The y-coordinate of the top left corner. [OUT] IBRX - The x-coordinate of the bottom right corner. [OUT] IBRY - The y-coordinate of the bottom right corner.
Returns	Nothing

GetWindowControlID([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns its control ID.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	The unsigned long representing the control ID.

GetWindowStyle([in]ULONG hWnd)

Description	For a given window, returns an unsigned long representing its window style. Note: Each bit of the return value represents a flag. Check Microsoft Windows style documentation to figure out the meaning of the bits.
Details	
Parameters	[IN] hWnd: The handle to the window as unsigned long.
Returns	An unsigned long representing the window style. In case the window is not found, the function returns 0.

GetWindowStyleEx([in]ULONG hWnd)

Description	For a given window, returns an unsigned long representing its extended window style. Note: Each bit of the return value represents a flag. Check Microsoft Windows style documentation to figure out the meaning of the bits.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	An unsigned long representing the windows extended style. In case the window is not found, the function returns 0.

GetWindowParent([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns its parent window.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	If the window is a child window, the return value is a handle to the parent window. If the window is a top-level window, the return value is a handle to the owner window. If the window is a top-level unowned window or if the function fails, the return value is 0.

GetWindowOwner([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns its owner window.
Details	
Parameters	[IN] hWnd: The handle to the window as unsigned long.
Returns	Get the owner window. In case the window has no owner or if the window handle is not correct.

GetWindowRoot([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns the root window by walking the chain of parent windows.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	The return value is the root window as unsigned long. If there is no root window, the function returns 0.

GetWindowClass([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns its window class. Note: This API throws an exception if the handle to the window is not correct. The class name can be 256 characters long.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	The string representing the window class.

GetWindowThreadID([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns its thread ID.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	The unsigned long representing the thread ID, if the window handle is not correct, then returns 0.

GetWindowProcessID([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns its process ID.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	The unsigned long representing the process ID, if the window handle is not correct, then returns 0.

GetWindowEnabled([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns whether it is enabled or not enabled.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	VARIANT_TRUE (-1) for enabled window, VARIANT_FALSE (0) for not enabled window or window handle is not correct.

GetWindowVisibility([in]ULONG hWnd)

Description	For a given window specified through its window handle, returns whether the window is visible. Note: The visibility state of a window is indicated by the WS_VISIBLE style bit. When WS_VISIBLE is set, the window is displayed and subsequent drawing into it is displayed, as long as the window has the WS_VISIBLE style. Any drawing to a window with the WS_VISIBLE style is not displayed if the window is obscured by other windows or is clipped by its parent window.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	VARIANT_TRUE (- 1) for visible window, VARIANT_FALSE (0) for visible window or window handle is not correct.

EnableWindow([in]ULONG hWnd, [in]VARIANT_BOOL bEnable)

Description	For a given window specified through its window handle, enables or disables the window.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long. [IN] bEnable - VARIANT_TRUE (-1) to enable, VARIANT_FALSE (0) to disable.
Returns	Nothing

ShowWindow([in]ULONG hWnd, [in]VARIANT_BOOL bShow)

Description	For a given window specified through its window handle, shows or hides the window.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long. [IN] bShow - VARIANT_TRUE (-1) to show, VARIANT_FALSE (0) to hide.
Returns	Nothing

CloseWindow([in]ULONG hWnd)

Description	For a given window specified through its window handle, closes the window. Note: The API throws an exception if the window cannot be closed because it is not correct or any other reason.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long to close.
Returns	Nothing

ClickWindow([in]ULONG hWnd, [in, optional]long nX, [in, optional]long nY)

Description	For a given window specified through its window handle, clicks the window at the optional position specified. Note: This API throws an exception for window handles that are not correct.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long. [IN] nX - The optional X coordinate specified relative to the top-left corner of the window. If not specified, middle of the window is assumed. [IN] nY - The optional Y coordinate specified relative to the top-left corner of the window. If not specified, middle of the window is assumed.
Returns	Nothing

FocusWindow([in]ULONG hWnd)

Description	For a given window specified through its window handle. Note: This API throws an exception for window handles that are not correct.
Details	The window is brought to focus by clicking its non-client area at position - 1, 5, and this API only works with top-level window which have a title-bar.
Parameters	[IN] hWnd - The handle to the window as unsigned long.

FocusWindow([in]ULONG hWnd)

Returns	Nothing
----------------	---------

SetWindowText([in]ULONG hWnd, [in]BSTR bstrText)

Description	For a given window specified through its window handle, tries and sets the specified text. Note: This API throws an exception for window handles that are not correct. The API calls the standard windows method of setting text and does not do keyboard simulation.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long. [IN] bstrText - The string to set.
Returns	Nothing

GetChildWindows([in]ULONG hWnd)

Description	For a given window specified through its window handle, gets all its immediate children windows. Note: This API throws an exception for window handles that are not correct.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long.
Returns	An array of unsigned long, each representing a child window handle.

EnableWindowMenu([in]ULONG hWnd, [in]BSTR bstrMenuPath, [in]VARIANT_BOOL bEnable)

Description	For a given window specified through its window handle, enables the menu item specified through the menu-path.
Details	
Parameters	[IN] hWnd - The handle to the window as unsigned long. [IN] bstrMenuPath - The menu path from the root (For example, File/Exit) [IN] bEnable - VARIANT_TRUE (-1) to enable, VARIANT_FALSE (0) to disable.
Returns	Nothing

GetListViewAccessor([in] ULONG hWnd,[out,retval] IListControl pListObj);**

Description	Provides an accessor object to get more details for a List-View control given the window handle to the list-view. Note: Use the accessor object to get information (such as the number of rows and columns), as well as what text is available at a given row and column combination (see the details of the ListControl accessor object).
Details	
Parameters	[IN] hWnd - The handle to the list view window.
Returns	A ListControl accessor object. If the window is not a list-view or if the window is not found, then the object returned is null.

SetWindowPosition([in] ULONG hWnd, [in] LONG nX, [in] LONG nY,[out,retval] int* bSuccess)

Description	Sets the position of a window to the location with respect to the client area of the parent window. Note: If the specified window is at the top level then the desktop is used as the parent to determine the location.
Details	
Parameters	[IN] hWnd - The handle to the window. [IN] nX - The x-position in client coordinates of the top-left corner of the window. [IN] nY - The y-coordinate in client coordinates of the top-left corner of the window.
Returns	1 for ok 0 for error

GetWindowPosition([in] ULONG hWnd, [out] VARIANT* nX, [out] VARIANT* nY, [out] VARIANT* cx, [out] VARIANT* cy)

Description	Get the position of a window with respect to the client area of the parent window. If the specified window is at the top level then the desktop is used as the parent to determine the location.
Details	
Parameters	[IN] hWnd - The handle to the window. [OUT] nX - The x-position of the top-left corner of the window. [OUT] nY - The y-position of the top-left corner of the window. [OUT] cx - The width of the window. [OUT] cy - The height of the window.
Returns	Nothing

SimpleClickWindow([in] ULONG hWnd);

Description	Posts a click message to a window. The window is sent a WM_LBUTTONDOWN and a WM_LBUTTONUP windows message, both pointing to the center of the window.
Details	
Parameters	[IN] hWnd - The handle to the window.
Returns	Nothing

GetFFBasicAuthServerLabel([in] ULONG hWnd, [out, retval] BSTR * label)

Description	Get the label string from the Firefox basic authentication dialog [in] hWnd: The handle to the basic authentication window.
Details	
Parameters	
Returns	The label text which can be used to extract the server name to determine the authentication service.

GetIAccessibleFromWindow([in] ULONG hWnd, [out, retval] IDispatch pRet)**

Description	Get the IAccessible interface from the window handle. This interface can be used to further query the window for information it might have made available for Accessibility support.
Details	
Parameters	[IN] hWnd - The handle to the window for which the IAccessible interface is fetched.
Returns	The IAccessible interface for the given window if available. Otherwise it returns NULL.

ListControl

Use these tables to view the plug-in API specifications for AccessStudio **ListControl**. This object is returned by the **GetListViewAccessor** method of the Window Controller.

The following are the methods exposed by this object:

GetColumnCount([out,retval] LONG* nCount)

Description	Get the number of columns in the list-view.
Details	
Parameters	
Returns	The column count.

HRESULT GetRowCount([out,retval] LONG* nCount)

Description	Get the number of columns in the list-view.
Details	
Parameters	
Returns	The row count.

GetAt([in] LONG nRow, [in] LONG nColumn, [out,retval] BSTR* pStr)

Description	Get the text at a given row and column combination. Note: This function can only be used for strings with a length less than 256 characters.
Details	
Parameters	[IN] nRow - The row for fetching the text. This parameter is a 1 based index (that is, pass 1 for the first row). [IN] nColumn - The column for fetching the text. This parameter is a 1 based index (that is, pass 1 for the first column).
Returns	The text at the given row and column location.

HRESULT GetSelectedItemIndex([out,retval] LONG* nIndex)

Description	Get the currently selected row in the list control.
Details	
Parameters	

HRESULT GetSelectedItemIndex([out,retval] LONG* nIndex)

Returns	The 1 based row number. If no row is selected, the function returns -1. if multiple items are selected, the function returns 0.
----------------	---

[id(6), helpstring("returns if an item at index (1 based) is selected")] HRESULT IsItemSelected([in] LONG nIndex,[out,retval] VARIANT_BOOL* pbSelected)

Description	Checks if a given row is selected.
Details	
Parameters	[IN] nIndex - The 1 based row number.
Returns	1 for selected, 0 for not selected.

Chapter 5. Troubleshooting AccessProfiles

You can use log files to trace back AccessProfile execution problems.

Playing back Observer log file

AccessStudio provides a playback function so you can trace recorded log events to troubleshoot the execution of AccessProfiles on other computer or device.

Before you begin

Ensure that the collected Observer log file contains at least log level 3 information. This log level setting provides more log information and contains SOAP logs.

To change the log level setting, do the following tasks:

1. Log on as an Administrator.
2. Go to **Registry Editor > HKEY_LOCAL_MACHINE > SOFTWARE > IBM > ISAM ESSO > ECSS > DeploymentOptions**.
3. Double-click **LogLevel** and type the log level. For more information on log level, see "Log policies" in the *IBM Security Access Manager for Enterprise Single Sign-On Policies Definition Guide*.

About this task

You can import one or more Observer log files and view the states, triggers, and actions that are processed in different applications in the AccessStudio **Real-Time Logs** pane. You can also view a trace of the log files in the state engine diagram.

Procedure

1. On the AccessStudio toolbar, type the path of the folder that contains the Observer log files that you want to use. You can also click **Browse** to locate an Observer log file.
2. Click **Load** to display the contents of the Observer log file in the **Real-Time Logs** pane. The logs are displayed on separate tabs for each process.
3. Click the tab of the process that you want to play back and click the **Start playback** icon.

In the AccessStudio **Real-Time Logs** pane, each log entry is highlighted for a period relative to the actual time it took to process the state, trigger, or action.

To view the trace in the state engine diagram, you must load the AccessProfiles used in the logs and click the **States** tab before you start the playback.

4. Click the **Pause playback** icon to pause the playback.
5. Click the **Next** icon to pause the playback and manually go to the next log entry.
6. Click the **Stop playback** icon to end the playback.

Note: You can view only one tab at a time. Stop the playback or wait for the process log to finish playing before you play another process log.

Chapter 6. AccessProfiles testing

Use the Test function of AccessStudio to verify whether the AccessProfiles applications are functioning correctly.

After creating AccessProfiles, AccessStudio displays tabs with process names. The tabs display the logs of all active applications that AccessProfiles defined for them. New tabs are created for each process id.

New tabs are created for each process id. For example, if you have two active applications on your computer with defined AccessProfiles - Patient Information Manager, and IBM Lotus® Sametime, the **Messages** pane displays separate tabs for each application.

When you start your test, launch the applications with the configured AccessProfiles in AccessStudio. The test is run for all AccessProfiles whose corresponding applications are active on the computer. A log is created for each one of these applications in addition to the existing logs.

For example, assume that two logs were displayed in the **Messages** pane that is based on AccessProfiles running on AccessAgent. These logs are for Patient Information Manager and IBM Lotus Sametime. Assuming that you configured AccessProfiles in AccessStudio for two other applications - *Messaging Software* and *e-mail Software*. You then start the test and launch these applications.

The **Real-Time Logs** pane displays four logs: Patient Information Manager, IBM Lotus Sametime, *Messaging Software* and *e-mail Software*. The first two are for AccessProfiles running on AccessAgent. The last two are for those logs which you are testing using AccessStudio. Close any of these using the **Close** button at the right corner of the pane.

Note: When you start a test using AccessStudio, AccessAgent Wallet is temporarily cleared until the test is stopped. This setting means that logon automation on your computer does not work until the time the test is stopped.

For more information:

- “Testing AccessProfiles”
- “Viewing log descriptions using a sample AccessProfile” on page 114

Testing AccessProfiles

Use the Test function to perform real-time tests on AccessProfiles.

About this task

This function is either accessible from the **Test** menu (**Test > Start**), or the **Test** icon from the toolbar.

Procedure

1. Open the Messages pane (**View > Messages**).
2. Press **F5** on your keyboard, or select **Test > Start** from the menu.
You can also click the **Start Test** or the **Restart Test** icon from the toolbar.

3. Launch the applications with AccessProfiles that you want to test.
 - For applications that are already active on your computer, restart the applications before you start the test. Verify that the application is closed before you restart it.
 - For applications such as virus scanners, access the **Services** console through **Administrative Tools** to completely shut down the application.
 - For applications such as Windows Explorer, you must ensure that the application .EXE process is ended by accessing the Windows Task Manager and navigating to **Processes**. Select **File > New Task (Run...)** from the menu and restart the application by typing the application .EXE name. You can also launch the application from the **Start** menu.

A log is created for each application, in addition to the existing logs in the Messages pane. The contents of these logs are in “Viewing log descriptions using a sample AccessProfile.”

4. Perform the relevant actions on the application to verify whether the AccessProfile is run correctly.
5. Stop the test by either pressing **Shift+F5** on your keyboard, go to **Test > Stop**, or click the **Stop Test** icon from the toolbar.

Results

The results of the test are provided in the **Messages** pane. A tab with a process name displays the logs of all active applications that have AccessProfiles defined for them. New tabs are created for each process ID.

Viewing log descriptions using a sample AccessProfile

This topic provides a description of the logs by using an AccessProfile of a sample application. You can also use this procedure to verify whether a sample application is working properly.

Before you begin

The Messages pane must be open.

About this task

The application logs on the **Messages** pane show the following details:

- The name of the application that is tested
- The triggers fired
- The transitions between states
- The actions that are run

Use this procedure to run a test for a sample application.

Procedure

1. Press **F5** on your keyboard to run the script.
2. Launch the application. A new tab with the name of the application you are testing is created for the application in the Messages pane.
3. Enter your user credentials in the **User name** and **Password** fields.
4. Click **OK**.

Note: Click the hyperlink to select or highlight the corresponding state, trigger, or action in the **States tab** diagram.

Chapter 7. Managing AccessProfiles

You can use AccessStudio to download, upload, or save information for AccessProfiles.

Downloading, uploading, and saving information for AccessStudio

Use AccessStudio to download AccessProfiles and associated information (which includes application objects, authentication services authentication service groups, and authentication service group links), from either the IMS Server or from the AccessAgent installed on your computer.

When you create information (like a new authentication service) or modify information, upload it to the IMS Server. This makes the new or updated information available to all AccessStudio users.

Use AccessStudio to save the AccessProfiles and the additional information you configure in a separate file.

For more information:

- “Downloading information”
- “Uploading information” on page 118
- “Saving information” on page 118

Downloading information

Use AccessStudio to download AccessProfiles and associated information. You can either download the information from the IMS Server, or from AccessAgent.

Downloading AccessProfiles and associated information from the IMS Server

Use AccessStudio to download AccessProfiles and associated information from the IMS Server.

Procedure

1. Click the **File** menu.
2. Select **Import data from IMS [your server name]**. This option downloads all existing AccessProfiles and associated information about the IMS Server.

Note: AccessStudio connects to the same IMS Server as configured for the AccessAgent on the computer. If download is successful, a confirmation note Loaded data from IMS shows in the status bar.

Downloading AccessProfiles and associated information from AccessAgent

Use AccessStudio to download AccessProfiles and associated information from the AccessAgent installed on your computer.

Procedure

1. Click the **File** menu.
2. Select **Import data from local AccessAgent**.

This option downloads all existing AccessProfiles and associated information from your local AccessAgent.

Results

When downloaded, you can view this information, modify it, or save it in a file for future reference and use.

What to do next

For the changes to be applicable globally, upload this information back to the IMS Server.

Uploading information

You can upload the following to the IMS Server after you create or modify them: AccessProfiles, application objects, authentication services, policies that are associated with the authentication services, authentication service groups, and authentication service group links.

Procedure

1. Select the AccessProfile or associated information from the Data type pane.
2. Click **Upload selected data to IMS** from the toolbar.

You can also right-click on the selected data and select **Upload to IMS**.

Saving information

Use AccessStudio to save AccessProfiles and other information you configure in a separate file. You must save the profile to a file before you edit or test a AccessProfile.

Procedure

1. Click the **File** menu.
2. Select **Save** or click the **Save selected data to file** icon.
3. Specify a name and location to save the file.
4. Click **Save**.

Results

The data that you selected and the dependent data are now saved to a file.

Chapter 8. Managing authentication services

Most applications require the validation of logon information by a verification entity. In AccessStudio, a reference is created to these entities through Authentication Services. AccessProfiles associated with the same authentication service belongs to the same verification entity. Changes that are made to the logon information in one AccessProfile are reflected across all others that are associated with the authentication service.

This concept is explained in “Associating authentication services with AccessProfiles.”

All these applications must share the set of credentials. If you associate applications with different sets of credentials with the same authentication service, it results in an error for the specific AccessProfiles. User credentials are stored in the Wallet according to the authentication service and not the application.

This information is useful to those using the **Form Editor** tab or **XML Editor** tab to create AccessProfiles.

For more information:

- “Associating authentication services with AccessProfiles”
- “Creating authentication services” on page 121
- “Modifying authentication services” on page 124
- “Creating authentication service groups and authentication service group links” on page 125

Associating authentication services with AccessProfiles

You can define authentication services in AccessStudio by using the **Authentication Services** function in the **View** menu.

At a minimum, you must provide an ID and a display name for the authentication service. Additional information is specified depending on your requirements. Authentication services can be associated with AccessProfiles in two ways: directly and indirectly.

Direct auth-info

Direct auth-info is a direct reference to an existing authentication service configured through the **Authentication Services** function in AccessStudio.

When you configure an authentication service as a direct reference, specify the authentication service ID and display name. This ID is displayed in the Form editor field when configuring an AccessProfile.

This is the direct reference. In most cases, a direct auth-info reference is sufficient.

Indirect auth-info

Indirect auth-info is used when you do not know which authentication service to select at the time of creating an AccessProfile. It is an indirect reference to an existing authentication service.

When you configure an authentication service as an indirect reference, in addition to the ID and display name, provide information about the server locators. Server locators help identify the entity that verifies the logon information for the user.

An example of a server locator for a Web site is *example.com*. This server locator information specifies information about the indirect reference. Indirect references are made to controls, elements, or windows of applications from which information is extracted and matched with the information specified for a server locator.

When a match is made, the authentication service is determined and associated with the application. In addition, you can specify a regular expression to further refine the information that is extracted from the indirect reference item.

Indirect auth-info references are useful in cases where you are unsure of what authentication service to use in your AccessProfiles.

The indirect reference process is illustrated in the following diagram.

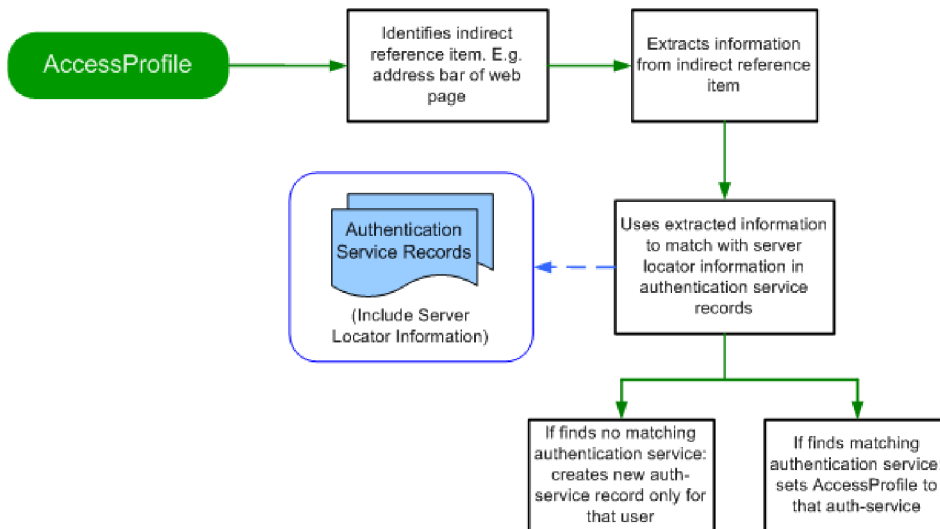


Figure 10. Indirect auth-info reference process

The diagram illustrates that when you run an AccessProfile, it identifies the indirect reference item that is specified in the AccessProfile. It then extracts information from the reference item.

For example, it can extract information from the address bar of a web page. It then tries to match the extracted information with the information specified for the server locators of existing authentication service records on the IMS Server.

If it finds a matching record, it sets the authentication service to the ID of the matching authentication service record.

If it does not find a matching authentication service record, it creates an authentication service record only for that user and creates an association with this record.

This new authentication service is not available for any other user or at the IMS Server for further association with new AccessProfiles.

Creating authentication services

Use AccessStudio to create an authentication service for an application.

Procedure

1. From AccessStudio, click the **New** icon and select **New Authentication Service**. A new authentication service is created (authenticator1), and the data is in the corresponding fields that are displayed on the **Form Editor** tab.
2. Specify information for the new authentication service.

Table 9. Authentication service data

Option	Description
Id	Enter a unique name to identify the authentication service. This field is mandatory for creating a direct auth-info reference.
Display name	Specify a name for the authentication service to be displayed in the AccessAgent Wallet. This field is mandatory for creating a direct auth-info reference.
Description	(Optional) Enter a short description for this authentication service.
Account data template id	You can also specify the account data template to use for the authentication service. An account data template defines the structure of the account data or user credentials. The default account data template ID is adt_ciuser_cspwd . This setting indicates the nature of user credentials user name that is not case-sensitive and a case-sensitive password.

Table 9. Authentication service data (continued)

Option	Description
Policies	<p>Expand this field group to define policy settings for an authentication service.</p> <p>These policies are the same policies that can be set in AccessAdmin.</p> <p>For enterprise authentication services, specify policy information. Mark the check box to enable the fields in this group.</p> <p>Set the default sign-on behavior for this authentication service by selecting from the list.</p> <p>The options are: Auto-logon, Always, Ask, and Never.</p> <p>Mark Require re-authentication before performing automatic sign-on if you want AccessStudio to authenticate the user for each automatic sign-on.</p> <p>Mark Prompt user on auto-capture of password to always ask the user whether to capture each password that is used for this authentication service.</p> <p>Specify restrictions for the password (such as minimum length, number of alphabetic characters, digits, and others).</p> <p>Enter a number or use the arrow keys in each field.</p> <p>Mark Enforce mixed-case passwords to require users to enter a combination of uppercase and lowercase characters for their passwords.</p> <p>After setting and testing all the authentication service policies, click the Upload selected data to IMS icon so the policies takes effect in AccessAdmin.</p>

Table 9. Authentication service data (continued)

Option	Description
<p>Dialog labels (Optional)</p>	<p>These labels are used when AccessAgent displays a dialog box for the user to enter their credentials.</p> <p>Click Add to add custom labels for authentication credentials.</p> <p>Specify labels only for account data items that are defined in this account data template of the authentication service. For example, for the adt_ciuser_cspwd account data template, specify labels for aditi_ciuser and aditi_cspwd.</p> <p>If a dialog box label is not specified, default labels, such as Username (case-insensitive) are used.</p>
<p>Localization support</p>	<p>Specify from the list, the culture or language for which support must be added.</p> <p>Specify the overriding name for this object if this culture is selected in AccessAgent or the IMS Server. This value is used for display purpose in the Wallet manager of the AccessAgent. The default value is used whether no override is found for a particular culture.</p> <p>Specify the overriding description for this object if this culture is selected in AccessAgent or the IMS Server.</p>

Table 9. Authentication service data (continued)

Option	Description
<p>Server Locators</p>	<p>For using indirect auth-info references, expand the Server Locators twistie.</p> <p>Specify server locator information for the applications you want to associate with this authentication service in the text box beside the Add button.</p> <p>For example, enter <code>www.xyz.example.com</code>.</p> <p>Click Add.</p> <p>You can add multiple server locators under one authentication service. This task is useful for multiple applications with the same authentication service, but with different server locators.</p> <p>For example, you can have a domain named <code>example.com</code>, and its subdomains use the same authentication.</p> <p>You can specify server locators for each of the services running on the subdomains (for example, <code>http://123.example.com</code> or <code>http://456.example.com</code>).</p> <p>You can also specify separate server locators for auto-fill of user credentials and that for credential capture. Clear the selection in the Use same server locators for injection and capture field to display separate server locator fields for auto-fill and capture.</p>

Note: By setting authentication service policies in AccessStudio, you can test the new policies that you created without affecting other AccessAdmin users.

3. Click the **Save** icon to store the new authentication service in an `.EAS` file.

Modifying authentication services

Use AccessStudio to modify the settings for an existing authentication service. Select the authentication service list item from the Data type pane and modify the information in **Form Editor** tab. If you are modifying the **Id** field, ensure that no AccessProfiles are currently associated with the authentication service, as it can cause errors with the AccessProfiles.

Managing authentication service groups and group links

Associating AccessProfiles with independent authentication services is typically sufficient. However, there are cases where even the user interface of an application cannot identify the authentication service. In such cases, you can create an **Authentication Service Group**, and associate multiple authentication services with this group.

Use the **Authentication Service Groups** function in AccessStudio to configure an authentication service group. When you create a group, you can link authentication

services with the group by using the **Authentication Service Group Links** function. You can then associate this group with an advanced AccessProfile. The associated user accounts are displayed to the user as logon options when the user accesses the application.

Tip: You can associate authentication service group only with advanced AccessProfiles.

Example

The following example is a scenario for creating authentication service groups and group links.

Assume that you have an application named ENC. The ENC application has two authentication services that validate user credentials - **auth_portal** and **auth_enterprise**. There is no existing means to determine which authentication service must be used for a specific user logon.

Create an authentication service group authgroup_ENC, and link the group to the two authentication services **auth_portal** and **auth_enterprise**.

Associate authgroup_ENC group with an advanced AccessProfile that you created for the ENC application.

When the user accesses the ENC application, the user is prompted to select the user account. Then AccessStudio selects the authentication service for the user to logon.

Creating authentication service groups and authentication service group links

Use AccessStudio to create authentication service groups and authentication service group link.

Procedure

1. From AccessStudio, click the **New** icon, then select **New Authentication Group**. The corresponding fields are displayed on the **Form Editor** tab.
2. Enter information for the new authentication service group.

Option	Description
Id	Enter a unique identification name for the group.
Add a server locator	<p>You can also create an indirect reference for the authentication service group as you did for an authentication service by specifying the server locator information.</p> <p>Specify server locator information for the applications you want to associate with this authentication service group in the text box beside the Add button. Click Add.</p> <p>You can add multiple server locators under one authentication service group. This feature is useful when you have multiple applications which use the same authentication service group but have different server locators.</p>

Option	Description
Server locators are case-sensitive	Specify whether the entered server locators are case-sensitive.

3. Create a link between the group and relevant authentication services.
Select **View > Advanced Data > Authentication Service Group Links** from the menu. The existing authenticator groups are displayed.
4. Click the **New** icon and select **New Authentication Group Link**.

Note: Authentication group links specifies which authentication service belongs to what group.

A new authentication group link is created (group_id1::auth_id1) is created. The corresponding fields are displayed on the **Form Editor** tab.

5. Select the **authentication service group link ID** from the Data type pane, then select the authentication service group ID to be linked with the authentication service from the **Authentication service group id** list.
6. Select the authentication service id of the authentication service to be linked with the authentication service group from the **Authentication service id** list.
7. Click the **Save** icon to store the new authentication service in an .EAS file.

Modifying authentication service group

If there are changes to the authentication service group, edit it through the Authentication Service Groups function

Procedure

Access the **Authentication Service Groups** function, select the authentication service group list item from the Data type pane and modify the information in the **Form Editor** tab.

Important: If you are modifying the **Id** field, ensure that no AccessProfiles are currently associated with the authentication service group, as it can cause errors with the AccessProfiles.

Note: If you change the **Id** field after uploading, delete the old authentication service name in the IMS Server. Otherwise, both the new and the old authentication service names are displayed in the IMS Server.

Modifying authentication service group links

Use AccessStudio to modify information for authentication service group links.

Procedure

1. Access the **Authentication Service Group Links** function, and select the authentication service group link list item from the Data type pane.
2. Modify the information in the **Form Editor** tab.

Chapter 9. Managing application objects

An application object in AccessStudio is a logical representation of a set of executable files (.EXE) or Web pages. It provides you with tighter control to apply policies on a group of AccessProfiles. Each AccessProfile must be associated with an application object. AccessProfiles can be associated with the same application object.

For more information:

- “Creating an application object”
- “Modifying an application object” on page 128

Creating an application object

Use AccessStudio to create an application object and associate it with an existing AccessProfile.

Procedure

1. From AccessStudio, click the **New** icon and select **New Application**. The corresponding fields on the **Form Editor** tab are displayed.
2. Enter information for the new application object.

Option	Description
Id	Modify the application ID based on the conventions of your organization.
Name	Enter the name for the application object. The name must be reflective of the group of .EXE files or web pages the application object is representing.
Description	Enter more information about the new application object.

Option	Description
<p>Policies</p>	<p>Expand this field group to specify default policy settings for the application object.</p> <p>Select a default action, when the user logs off from AccessAgent.</p> <p>The options are:</p> <ul style="list-style-type: none"> • Log off the application • Close the application • Do nothing <p>Select an automatic sign-on option for the application object from the list.</p> <p>The options are:</p> <ul style="list-style-type: none"> • Automatic logon • Always, Ask • Never • Certificate <p>After setting up and testing all the authentication service policies, click the Upload selected data to IMS icon so the policies takes effect in AccessAdmin.</p>
<p>Localization support</p>	<p>Specify from the list the culture or language for which support must be added.</p> <p>Specify the overriding name for this object if this culture is selected in AccessAgent or the IMS Server. This value is used for display purposes in the Wallet manager of the AccessAgent. The default value is used whether no override is found for a particular culture.</p> <p>Specify the overriding description for this object if this culture is selected in AccessAgent or the IMS Server.</p>

Note: By setting application object policies in AccessStudio, you can test the new policies that you created without affecting other AccessAdmin users.

3. Click the **Save** icon to store the new application object in an .EAS file.
4. After creating the application object, upload it to the IMS Server where AccessAgent is connected.

The application object can then be available for all users.

To upload to the IMS Server, right-click on the application object and click **Upload to IMS**. A message displays indicating the success or failure of the upload.

Modifying an application object

Use AccessStudio to modify information in an application object that is associated with an existing AccessProfile.

Procedure

1. Select the application object in the Data type pane.
2. Edit the details for the application object in **Form Editor** tab.

Important: If you are modifying the **Id** field, ensure that no **AccessProfiles** are currently associated with the application object, as it can cause errors with the **AccessProfiles**.

Chapter 10. Account data items and templates

Account data represents user logon information in AccessStudio, such as the user name and password. The account data for an AccessProfile is stored in a specific format that is defined in the account data templates.

Account data templates include individual account data items. The properties of these items are defined in the account data item templates. These templates are accessible in AccessStudio through the account data templates and account data item template functions.

The templates are predefined in AccessStudio. You can view the templates using the respective functions, but you cannot modify the template.

For more information:

- “Account data item templates”
- “Account data templates”
- “Viewing account data item templates or account data templates” on page 133

Account data item templates

An account data item template defines the properties of individual account data items. Account data are the user credentials that are required for logon.

An account data item template defines whether the field entry is:

- A secret field that requires encryption
- Case sensitive

The template also defines the display name and description of the field entry. The field name and description are available in multiple languages. It is displayed in the language that is specified by the user during AccessAgent installation.

User name fields are typically not secret and are not case sensitive. For **Password** fields, they are typically secret and are case sensitive.

Account data items are identified through IDs in AccessStudio. Each ID is structured so that it instantly provides you with the properties of the account data item.

For example, **aditi_cspwd** is broken into two parts: **aditi** and **cspwd**. **aditi** is the "account data item template information". **cs** in the **cspwd** means "case sensitive", and **pwd** means "password". So, this account data item ID indicates that the account data item is a case-sensitive password field.

Account data templates

Account data templates define the format of account data to be stored for credentials that are captured by using a specific AccessProfile. The account data name and description are available in multiple languages. It is displayed in the language that is specified by the user during an AccessAgent installation.

An account data template includes:

- **Account data item templates**
- **Key identifier (identifies whether the account data item is key or non-key)**

A key account data item indicates that it is a unique identifier for that set of account data. Information that is captured for the key account data item cannot be changed. For a non-key item, information is considered non-constant and can be changed within that set of account data.

For example, a user name can be considered as the key item for the account data and the password the non-key. The password for that user name can change as many times as required. However, the password retains its identity as part of the account data with the user name.

The credentials are regarded as part of a different data if the user name changes, whether the password is changed or not.

Account data templates are identified through IDs in AccessStudio. Each ID provides you with the properties of the account data items in the template.

For example, **adt_csuser_cspwd** can be divided into three parts: **adt**, **csuser**, and **cspwd**. **adt** means 'account data template'. **csuser** means 'case-sensitive user name', and **cspwd** means 'case-sensitive password'. This account data template ID indicates that it contains two account data items - a case-sensitive user name and a case-sensitive password.

Account data templates can be associated with AccessProfiles and authentication services. When associated, the account data template is used by the account data bag of each AccessProfile.

An account data bag is a temporary data holder or cache that stores user credentials after their capture from the application screen, and before auto-fill.

The credentials are retrieved from the Wallet and stored in the account data bag before they are automatically inserted on the application screen fields. The user credentials are also stored in the account data bag after capture before they are transferred to the Wallet of the user.

The account data template information is extracted either from the AccessProfile or the associated authentication service when an AccessProfile is run. This template is then used to set the structure of the account data bag.

For example, the account data template **adt_csuser_cspwd** contains two account data items: a case-sensitive user name and a case-sensitive password. A structure is created for the account data bag, which includes two slots. The first slot captures a case-sensitive user name. The second slot captures a case-sensitive password.

Store the information in the corresponding fields. Then, associate an account data item template user name and password fields when you configure them in the AccessProfile.

In this example, users associate the **aditi_csuser** account data item template with the **user name** field of the application, and the **aditi_cspwd** account data item template with the **password** field. A simple match is conducted to identify data from which field belongs to which slot in the account data bag.

Viewing account data item templates or account data templates

Use AccessStudio to view the details for an account data item template or an account data template.

Procedure

1. Select **View > Advanced Data > Account Data Templates or Account Data Item Templates** from the menu.
2. Select the account data templates or account data item templates you want to view.

Chapter 11. Backing up IMS Server data

Use AccessStudio to make a backup of AccessProfiles and associated information that is stored on your IMS Server.

About this task

When you back up IMS Server data, AccessStudio downloads AccessProfiles, Authentication Service Applications, Authentication Groups, Authentication Group links, and Account data templates. It saves them in a custom AccessStudio .EAS file or .XML file (depending on your preferences).

Procedure

1. Select **Tools > Backup System Data from IMS to File** from the menu. A message is displayed.
2. Click **Backup**. You are prompted to provide information about the file name and format.
3. Select the file format from the **Save as type** list. You can either save the information in a .XML file or a custom AccessStudio .EAS file.
4. Provide a file name, and choose the location to save the file.

Results

A message is displayed, confirming the success of the backup.

Chapter 12. Triggers and actions

A trigger fires an action as a response, after a condition is met.

For example, after a web page completes loading, inject a credential.

Trigger: After a web page completes loading.

Action: Inject a credential.

Triggers

The table lists the predefined triggers in AccessStudio and explains the conceptual information for each trigger in detail with relevant examples.

Table 10. AccessStudio Triggers

Trigger	Description
Window is activated	<p>Windows raises an <i>activate</i> event on a window with a title bar when it is activated by the user or a program, by using the mouse or the keyboard, or while coding.</p> <p>This trigger is fired when the activation of the window occurs. The activation is characterized by the appearance of the window in the foreground with a highlighted title bar.</p> <p>The parameter of this trigger is the signature of the window that is activated.</p> <p>Example:</p> <p>To trap activate on window with title, "Messenger! signin",</p> <pre><wnd_activate_trigger> <signature>/child::wnd[@title="Messenger! signin"] </signature> </wnd_activate_trigger></pre> <p>Note: The window activate trigger works only on a window with a title bar. It works only for windows within the same process the agent is residing. The agent loads into the intended process before using activate trigger on the window within that process. Unlike other triggers, this trigger would not fail even if the window is not present when the trap is being set.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
<p>Button is clicked</p>	<p>Windows raises a <i>button click</i> event when a button is clicked on a window.</p> <p>This trigger is fired when this clicking of button occurs. The parameters for this trigger are signatures of the window and the button.</p> <p>Example:</p> <p>To trap the click of a button with control ID "101" within a window with the signature "/child:wnd[@title="Messenger"]", use</p> <pre data-bbox="719 600 1414 785"> <wnd_command_bn_click_trigger> <signature>/child:wnd[@title="Messenger"]</signature> <control> <signature>/child:wnd[@title="Messenger"]/ child:wnd [@ctrl_id="101"]</signature> </control> </wnd_command_bn_click_trigger> </pre> <p>Note: If the window with the specified signature is not found, no event trapping is set. To make sure that the window is available, set a trap for activate at the top-level parent window containing that window.</p>
<p>Web page completes loading</p>	<p>A <i>document complete</i> event takes place when a web page is fully loaded in the web browser. The status bar of the browser indicates that a web page is downloaded and displayed.</p> <p>This trigger is fired when loading is complete. This trigger is typically the first trigger in the begin state in an AccessProfile for a website.</p> <p>The parameter for this trigger includes the signature of the completed web page.</p> <p>When this trigger is run, other triggers that are associated with the specific web page (identified by using the signature) are run.</p> <p>Tip: To match dynamic pages with a single URL, you can specify the signature to a unique element on a page.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
HTML element is clicked	<p>A web browser raises a <i>click</i> event when an HTML element is clicked manually or automatically.</p> <p>This trigger fires when the HTML element is clicked.</p> <p>The parameter of this trigger is the signature of the HTML element which is being monitored for clicking.</p> <p>Example:</p> <p>To trap a click event on an image element which source is someimage.jpg, use:</p> <pre data-bbox="751 596 1390 699"><web_click_item_trigger> <signature>/child::html/child::html/child::html[@tag_name="img" and src="someimage.jpg"]</signature> </web_click_item_trigger></pre> <p>Note: If the HTML element to which the click event is to be trapped has a JavaScript for that event, it can mean either of the following events when trapping the event: after or before the JavaScript execution. Set the <code>before_javascript</code> attribute to 0 to trap after the script execution.</p>
Window gets created	<p>Windows raises a <i>create</i> event when a window is created.</p> <p>This trigger fires when this creation of a window occurs.</p> <p>The parameter for this trigger is the signature of the window which is being created.</p> <p>Example:</p> <p>To trap the creation of a window that would match / child::wnd[@title="Messenger"], use</p> <pre data-bbox="751 1192 1401 1272"><wnd_create_trigger> <signature>/child::wnd[@title="Messenger"]</signature> </wnd_create_trigger></pre> <p>Note: Unlike other triggers, this trigger would not fail if the window is not present while the trap is being set.</p>
Window gets destroyed	<p>Windows raises an event when a window is closed.</p> <p>This trigger fires when the window closes.</p> <p>The parameter of this trigger is the signature of the window to be closed.</p> <p>Example:</p> <p>To trap a destroy event on a window that says Messenger! Messenger with Voice, the signature would look like / child::wnd[@title="Messenger! Messenger with Voice" and @class_name="MessengerBuddyMain"]</p> <p>Tip: This trigger can be used to transit the state engine if a mouse event or a key press event cannot be captured in an application that has an owner-drawn window.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
Window is hidden	<p>Windows raises the <i>hide window</i> event on a window before hiding it.</p> <p>This trigger is fired when event is raised.</p> <p>The parameter of this trigger is the signature of the window which is to be hidden.</p> <p>Example:</p> <p>To trap hide window event on a window with text "Sign in"</p> <pre><wnd_hide_window_trigger> <signature>/child::wnd[@title="Sign in"]</signature> </wnd_hide_window_trigger></pre> <p>Note: If the window with a specified signature is not found, no event trapping would be set. To make sure that the window is available, set a trap for activate at the top-level parent window containing that window.</p>
Window is found	<p>This trigger searches for and fires when it finds the window specified in the signature parameter by polling that window.</p> <p>The other parameters to this trigger are the polling interval in seconds and the number of times the window is searched for, the polling count.</p> <p>Example:</p> <p>In some cases, the Window is found trigger is the only choice because the Window is activated trigger is unable to trap the activate event generated by Windows. In such cases, the Window is found trigger is a direct substitute for the Window is activated trigger.</p> <p>Note: Avoid using this trigger unless it is necessary. In general, any polling mechanism introduces a performance overhead that might degrade the user experience.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
<p>Key is pressed down on a window</p>	<p>Windows generates a <i>key down</i> event when a key or a combination of keys on the keyboard are pressed.</p> <p>The parameters of this trigger are the signature of the window that must be monitored for keyboard input, the key combination that match, and an optional regular expression if a specified text output is to be matched.</p> <p>Note: When you set the property Fire trigger on key to Any key, the Enter key does not cause the trigger to fire.</p> <p>Example:</p> <p>To capture the user name and password when logging in to <i>Messenger! Messenger</i>, specify the signature of the password field in the signature box and choose to fire the trigger on the Enter key.</p> <p>For a terminal application, a sequence of keystrokes - such as a command like change password - can be captured using the Key is pressed down on a window trigger and matched against the regular expression specified in the regex parameter of the trigger.</p> <p>A successful match can be used to trigger the change password workflow in the state engine. The regular expression look like <code>.*change.*</code></p> <p>Note: This powerful trigger monitors user input and can be used in state engines that perform logon automation on terminal or mainframe applications that need heavy user interaction.</p> <p>This trigger is sensitive to the signature of the window that is monitored for keyboard input.</p> <p>If there are problems detecting the keyboard input on a window, try making the signature more generic or more specific.</p> <p>If input from a previous Simulate keyboard input action is to trigger a Key is pressed down on a window trigger, ensure that the signatures for the input action and the key down trigger are identical.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
<p>Left mouse button is clicked</p>	<p>Windows raises a <i>left button down</i> event when the left button of a mouse is clicked in an application window.</p> <p>This trigger fires when this clicking of the left mouse button occurs.</p> <p>The parameter of this trigger is the signature of the window in which the left mouse button is clicked.</p> <p>Example:</p> <p>To trap the left mouse button click OK, the Signature element would be</p> <pre data-bbox="722 625 1166 730">/child::wnd[@title="Messenger Talk"]/ child::wnd[@class_name="Main View"]/ child::wnd[@class_name="#32770"]/ child::wnd[@class_name="Button" and @ctrl_id=1]</pre> <p>Tip: This trigger would normally contain a <code>capture_action</code> to capture the user name and password after the user clicks OK.</p>
<p>Right mouse button is clicked</p>	<p>Windows raises a <i>right button down</i> event when the right button of a mouse is clicked in an application window.</p> <p>This trigger fires when this clicking of the right mouse button occurs.</p> <p>The parameter of this trigger is the signature of the window in which the right mouse button is clicked.</p> <p>Example:</p> <p>To trap the right mouse button click an OK button, the Signature element would be</p> <pre data-bbox="722 1192 1372 1297">/child::wnd[@title="MessengerTalk"]/child::wnd[@class_ name="Main View"]/child::wnd[@class_name="#32770"] /child:: wnd[@class_name="Button" and @ctrl_id=1]</pre>
<p>MDI child window is activated</p>	<p>Windows raises an <i>activate</i> events for other applications when a child window of an application with a title bar receives focus or is selected by the user.</p> <p>The only parameter of this trigger is the signature of the child window that receives focus or is selected.</p> <p>Example:</p> <p>Some applications have child windows that can be monitored for <code>mdi_activate</code> events only. In such cases, the MDI child window is activated trigger is used instead of the Window is activated trigger.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
<p>Cursor moves on a window</p>	<p>Windows raises a <i>set cursor</i> event for a window on which the cursor is active.</p> <p>This trigger fires when the cursor moves on the specified window.</p> <p>The parameter of this trigger is the signature of the window which set cursor" event is trapped.</p> <p>Example:</p> <p>To trap a "set cursor" event on window with signature / child::wnd[@title="Messenger"]/child::wnd[@ctrl_id="102"], use</p> <pre data-bbox="751 653 1458 762"><wnd_set_cursor_trigger> <signature>/child::wnd[@title="Messenger"/ child::wnd [@ctrl_id="102"]</signature> </wnd_set_cursor_trigger></pre>
<p>Windows field gets focus</p>	<p>Windows raises the <i>set focus</i> event when any input field on the window is clicked.</p> <p>This trigger is fired when clicking occurs.</p> <p>The parameter of this trigger is the signature of the window whose input field that is being monitored for input field click.</p> <p>Example:</p> <p>To trap a "set focus" event on window with signature / child::wnd[@title="Messenger"]/child::wnd[@ctrl_id="102"], use</p> <pre data-bbox="751 1140 1458 1224"><wnd_set_focus_trigger> <signature>/child::wnd[@title="Messenger"/ child::wnd [@ctrl_id="102"]</signature></pre>
<p>Text is displayed on a window</p>	<p>Windows raises a <i>set text</i> event when the text of a window is modified by an external program.</p> <p>This trigger is fired when this modification by an external program occurs.</p> <p>The parameter of this trigger is the signature of the window where the text modification is to occur.</p> <p>Example:</p> <p>To trap a set text event on a window with signature / child::wnd[@title="Messenger"]/child::wnd[@ctrl_id="102"] use</p> <pre data-bbox="751 1629 1458 1734"><wnd_set_text_trigger> <signature>/child::wnd[@title="Messenger"/ child::wnd [@ctrl_id="102"]</signature> </wnd_set_text_trigger></pre>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
Window is shown	<p>Windows raises a <i>window show</i> event before a window is displayed on screen or before it is hidden (through minimization).</p> <p>This trigger fires before the window is displayed or hidden.</p> <p>The parameter of this trigger is the signature of the window which is to be displayed or hidden.</p> <p>Example:</p> <p>To trap the show event on a window with a text that says, "Sign in".</p> <pre data-bbox="722 625 1347 703"><wnd_show_window_trigger> <signature>/child::wnd[@title="Sign in"]</signature> </wnd_show_window_trigger></pre> <p>Note: If the window with the specified signature is not found, no event trapping would be set. To make sure that the window is available, set a trap to activate the top-level parent window containing that window.</p>
Window position changes	<p>Windows raises an event when the position of a window is changed on the desktop.</p> <p>This trigger fires when this change in window position occurs.</p> <p>The parameters to this trigger are the signature of the window that is to be monitored for position change and an optional flag parameter value (the flag is a member of the WINDOWPOS structure which can be obtained using Spy).</p>
Menu item is clicked	<p>This trigger fires when a specified menu item on the Windows application is clicked.</p> <p>The other parameters to this trigger are the signature of window with the menu and the path of the menu item.</p> <p>Example:</p> <p>For the trigger to fire Menu item is clicked, specify: Signature of the window with the menu:</p> <pre data-bbox="722 1413 1096 1522">/ child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"] Menu Path: File/Login</pre> <p>Tip: You can specify a multi-level menu item by providing a separator '/' in the menu path. For example File/Open.</p>
Text is found on a console window	<p>This trigger searches for and fires when it finds the text that is specified in the sections parameter by polling for text in the console application screen.</p> <p>The other parameters to this trigger are the sections of the text to match against the screen contents, the polling interval in seconds and the number of times the window is searched, the polling count.</p> <p>Note: As this trigger is a poller, use this trigger judiciously.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
Browser starts navigating	<p>A before <i>navigate</i> event takes place when the web browser starts navigating to a URL. This trigger is fired when this navigation to a URL begins.</p> <p>Example:</p> <p>To trap the event, user navigating out of a web page - for example, <code>http://www.mail.example.com/</code> - use:</p> <pre><web_before_navigate_trigger> </web_before_navigate_trigger></pre> <p>Tip: As this trigger would trap all before navigate events, use test property for filtering.</p>
HTML element completes loading	<p>The web browser raises an <i>on load</i> event when the loading of specific HTML elements on a web page is complete.</p> <p>For example, this event and trigger are fired when the body of an HTML page finishes loading.</p> <p>The parameter of this trigger is the signature of the HTML element which is to complete loading for the trigger to fire.</p> <p>Example:</p> <p>To trap an onload event on the body element,</p> <pre><web_elem_onload_trigger> <signature>/child::html[@tag_name="body"]</signature></pre>
Key is pressed on a web page	<p>A web browser raises the <i>key press</i> event when a key is pressed for an HTML element.</p> <p>This trigger fires when this event is raised for the specified HTML element.</p> <p>The parameter of this trigger is the signature of the HTML element which is being monitored for pressing of the key.</p> <p>Example:</p> <p>To trap a key press event on an image element which source is <code>someimage.jpg</code>, use</p> <pre><web_key_press_trigger> <signature>/child::html/child::html/child::html[@tag_name="img" and src="someimage.jpg"]</signature> </web_key_press_trigger></pre> <p>Note: If the HTML element of which the key press event has to be trapped has a JavaScript for that event, it means either of the following events: after or before the JavaScript execution. Set the <code>before_javascript</code> attribute to 0 to trap after the script execution.</p>
Browser closed	<p>A <i>browser close</i> event takes place when the web browser closes. This trigger is fired when this event takes place.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
Left mouse button is clicked on a web page	<p>Fires when the left mouse button is clicked on a web page control.</p> <p>Use this trigger for Web elements if the HTML element is clicked trigger does not work. This trigger always fires before any JavaScript handler on the web page.</p> <p>Provide the signature for the web element on which the mouse click has to be captured.</p> <p>This trigger cannot be used as the first trigger to kick off the state-machine in the AccessProfile.</p> <p>The HTML element is clicked trigger can fail for various reasons: either the element does not generate a click event, or the JavaScript code of the web page does not handle the click properly. Use this trigger in such cases.</p>
HTML element is found	<p>Fires when the specified HTML control is found.</p> <p>This trigger sets up a polling mechanism and searches for the specified HTML control.</p> <p>Specify the HTML control signature, the polling interval, and polling count. The first poll is done immediately, and subsequent polls are made at the specified interval.</p> <p>Note: Do not use this trigger as the first trigger to kick off the state-machine in the AccessProfile. Precede the first instance of this trigger with the Web page completes loading trigger.</p>
HTML element gained focus	<p>Fires when the specified HTML control gets keyboard focus.</p> <p>A flashing caret represents keyboard focus for an edit control.</p> <p>Use this trigger in scenarios when the web page modifies the contents of the edit control on gaining focus.</p>
HTML element lost focus	<p>Fires when the specified HTML control loses keyboard focus.</p> <p>A flashing caret represents keyboard focus for an edit control.</p> <p>Use this trigger in scenarios when the web-page modifies the contents of the edit control on losing focus.</p>
Text is first displayed (Terminal)	<p>A terminal application raises the <i>terminal application screen output</i> event when there is text output on the application screen.</p> <p>This trigger fires when this text output occurs. The parameter of this trigger is a regular expression of the text that is being searched.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
Text is first displayed (Mainframe)	<p>AccessAgent raises the <i>screen text change</i> event when there is a change in the text anywhere on a window of the process for which an AccessProfile is running.</p> <p>This trigger fires when this text change occurs.</p> <p>The parameter of this trigger is sections of the text to match against the window contents.</p> <p>Example:</p> <p>To trap a "screen text change" event when a window 10th line changes to "user name:"</p> <pre data-bbox="751 625 1174 884"><mf_screen_output_ex_trigger> <sections> <section> <match_text_section> <line from="10" to="10"></line> <match_text>user name:</match_text> </match_text_section> </section> </sections> </mf_screen_output_ex_trigger></pre>
Text is found (Mainframe)	<p>Specifies information for the trigger to fire when there is a specific text output on a mainframe application screen.</p> <p>A mainframe application raises the <i>screen output</i> event when there is text out anywhere on a mainframe application window.</p> <p>This trigger fires when this text output occurs.</p> <p>The parameter of this trigger is sections of the text to match against the window contents.</p> <p>Note: As this trigger is a poller, use this trigger judiciously.</p>
Text is displayed (HLLAPI)	<p>The HLL framework of an HLL-compatible Mainframe or terminal application generates a message when the application outputs some text to screen.</p> <p>This trigger is fired when this output occurs. Specific sections of text output to the screen can be matched by this trigger.</p> <p>The parameters of this trigger are the signature of the HLL-enabled application window and a regular expression of the text that is being searched.</p> <p>Note: This trigger works only for an application that is HLLAPI-enabled. Common HLLAPI-enabled applications are: Attachmate EXTRA!, IBM iSeries and Reflection.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
HLLAPI session starts	<p>The framework of an HLL application generates a message when an HLL-compatible Mainframe session starts.</p> <p>This trigger fires when this session starts.</p> <p>The parameters to this trigger are the signature of the HLL-enabled application window and the HLLAPI short name for the application.</p> <p>Example:</p> <pre data-bbox="719 541 1357 699"><hll_session_start_trigger> <short_name>A</short_name> <long_name></long_name> <wnd_signature>/child:wnd[@class_name="SDIMainFrame" and @title#"WS - EXTRA! X-treme"]</wnd_signature> </hll_session_start_trigger></pre> <p>Note: If the short name is not configured on the EXTRA application, then this trigger does not fire.</p>
Fire immediately	<p>This trigger fires at a zero second timeout. There are no parameters for this trigger.</p> <p>Example:</p> <p>This trigger is used to test the value of properties that were set in previous states.</p> <p>Note: Use two Fire immediately triggers in a state to control the flow in a state-engine by testing for one possible value of a Boolean condition as a test_property in each trigger.</p> <p>Do not use two Fire immediately triggers in the same state without test_properties transiting to different states. The behavior of such usage cannot be predicted reliably.</p>
On logon to AccessAgent	<p>AccessAgent raises a <i>logon</i> event when an AccessAgent user logs on.</p> <p>This trigger is fired when this logon occurs.</p> <p>The parameter of this trigger is the signature of the window on which automated logon is performed.</p> <p>Tip: This trigger is used to perform automated logon to an application that runs without being terminated across multiple user sessions, as soon as a user logs on to AccessAgent.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
On logoff from AccessAgent	<p>AccessAgent raises a <i>logoff</i> event when an active AccessAgent user logs off.</p> <p>This trigger is fired when logoff occurs.</p> <p>The parameters of this trigger are: Timeout in seconds, whether logoff is synchronized across applications, and whether logoff can happen concurrently. The default settings suffice for most cases.</p> <p>Example:</p> <p>This trigger is used to perform graceful logoff from desktop applications when a user logs off from AccessAgent.</p> <p>The actions in this trigger can save the current work of the user and log the user off or terminate the application when a logoff event is received.</p> <p>Tip: Use On logoff from AccessAgent triggers to ensure that a user is logged off an application regardless of when the user chooses to log off from AccessAgent. There are typically multiple On logoff from AccessAgent triggers in multiple states in a state engine.</p>
Fire after specified time	<p>This trigger always fires after the timeout counter expires. The parameter to this trigger is the timeout value, in seconds.</p> <p>Example:</p> <p>This trigger can be used to test the value of properties that were set in previous states.</p> <p>Use this trigger to wait for an application window to stabilize if the control IDs of the user name or password field change after window creation.</p> <p>Use this trigger as a last resort, if all the triggers in a particular state never fire. This trigger prevents the state engine from freezing in that state.</p> <p>Tip: Avoid using too many timeout triggers in an AccessProfile. Using too many timeout triggers makes the state-engine more difficult to troubleshoot or debug. It also increases the scope for user intervention during logon automation.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
<p>Java window is activated</p>	<p>JVM raises an <i>activate</i> event on a Java window with a title bar when it is activated by the user or a program, using the mouse or the keyboard, or programmatically.</p> <p>This trigger is fired when this activation of Java window occurs. The activation is typically characterized by the appearance of the Java window in the foreground with a highlighted title bar.</p> <p>The parameter of this trigger is the signature of the Java window that is activated.</p> <p>Example:</p> <p>To trap activate on Java window with title say, "Login",</p> <pre data-bbox="719 653 1333 730"><jwnd_activate_trigger> <signature>/child::jwnd[@title="Login"]</signature> </jwnd_activate_trigger></pre> <p>Note: The Java window activate trigger works only on a Java window with a title bar. It works only for windows within the same process the agent is residing. First, the agent loads into the intended process before using activate trigger on the window within that process. Unlike other triggers, this trigger would not fail even if the window is not present when the trap is being set. If this trigger is used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.</p>
<p>Java window is clicked</p>	<p>Java generates the <i>click</i> event when a Java window is clicked manually or automatically.</p> <p>This trigger fires when the Java window is clicked.</p> <p>The parameter of this trigger is the signature of the Java window which is being monitored for clicking.</p> <p>Example:</p> <p>To trap a click event on a Java window, use</p> <pre data-bbox="719 1339 1344 1577"><jwnd_click_item_trigger><signature>/ child::jwnd[@class_name="LoginPanel1"]/ child::jwnd[@class_name="javax.swing.JRootPane"]/ child::jwnd[@class_name="javax.swing.JLayeredPane"]/ child::jwnd[@class_name="javax.swing.JPanel"]/ child::jwnd[@class_name="javax.swing.JPanel"]/ child::jwnd[@class_name="javax.swing.JButton" and @is_visible=1 and @title="jButton1"]</signature> </jwnd_click_item_trigger></pre> <p>Note: If this trigger is used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
Java window is found	<p>This trigger is sensitive to the signature of the window that is monitored for keyboard input.</p> <p>If there are problems detecting the keyboard input on a window, try making the signature more generic or more specific.</p> <p>If input from a previous Simulate keyboard input action triggers a Key is pressed on a Java window trigger, ensure that the signatures for the input action and the key down trigger are identical.</p> <p>Note: Avoid using this trigger unless it is necessary. In general, any polling mechanism introduces performance overheads that spoil the user experience. If this trigger is to be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.</p>
Key is pressed on a Java window	<p>Java generates a <i>key down</i> event when a key or a combination of keys on the keyboard are pressed.</p> <p>The parameters of this trigger are the signature of the Java window that must be monitored for keyboard input and the key combination that match.</p> <p>Example:</p> <p>To capture the user name and password when logging in to an internet banking system, specify the signature of the password field in the signature box and choose to fire the trigger on the Enter key.</p> <p>Note: If this trigger must be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.</p> <p>This trigger is sensitive to the signature of the window that has to be monitored for keyboard input.</p> <p>If there are problems detecting the keyboard input on a window, try making the signature more generic or more specific.</p> <p>If input from a previous Simulate keyboard input action triggers a Key is pressed on a Java window trigger, ensure that the signatures for the input action and the key down trigger are identical.</p>

Table 10. AccessStudio Triggers (continued)

Trigger	Description
JVM becomes available	<p>This trigger fires when the JVM (Java Virtual Machine) is available.</p> <p>The other parameters of this trigger are the signature of window with the menu and the path of the menu item.</p> <p>Example:</p> <p>For the trigger to fire Menu item is clicked, specify: Signature of the window with the menu: <code>/child:wnd[@title="Form1" and @class_name="ThunderRT6FormDC"]</code> Menu Path: File/Login</p> <p>Note: For Java applications, this trigger is provided in the start state so that the JVM is ready before any automation or SSO starts.</p> <p>If this trigger must be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.</p>

Actions

See the table for information about the predefined actions in AccessStudio. This chapter explains the conceptual information for each action in detail with relevant examples.

Table 11. AccessStudio Actions

Action	Description
Inject credentials	<p>Enables automatic insertion of user account data (credentials) on corresponding logon fields in an application window.</p> <p>The action attempts to fetch user account data from the specified authentication service, and then transfers the account data into the inject account data bag.</p> <p>Finally, it inserts the user account data into the respective logon fields that are defined in the sso_items option.</p> <p>Note: If you have multiple logon screens for an application, you can use different injection bags for different authentication services.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Capture credentials	<p>Enables capturing of user credentials from the logon fields on an application window. These logon fields are the user name and password fields.</p> <p>This action is defined when enabling logon automation for any application.</p> <p>The captured credentials are stored in an account data bag, and then transferred to the Wallet after running a Save credentials action.</p> <p>This action runs after the Enter key is pressed for a credential field or after the left mouse button is clicked on an OK or Log On button.</p> <p>For more information, see the Key is pressed down on a window and Left mouse button is clicked triggers.</p> <p>For a Web application, this action can be contained inside an HTML element is clicked trigger or a Browser starts navigating trigger.</p> <p>Note: If an application is expected to run across multiple user or AccessAgent sessions, ensure that the Empty Account Data Bag First field value is set to Yes so that any cached credentials that belongs to the previous user are erased.</p>
Save credentials	<p>Saves the user account data (credentials) from the specified account data bag to the Wallet of the user.</p> <p>Tip: The action is typically used with the Account Data Capture action to save the account data from the capture bag to the Wallet. Ensure that valid account data is available in the bag before saving.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Transfers data	<p>Transfers data from a source location to a destination location. To specify data transfer information, right-click on the data_transfer_items and select Add Data-Transfer-Item.</p> <p>Next, right-click on the from and to the data items to specify the source and destination of the data to transfer.</p> <p>Example:</p> <p>Transfer of data from a window signature to a property:</p> <pre data-bbox="683 541 1419 989"> <data_transfer_action> <data_transfer_items> <data_transfer_item> <from> <wnd_xpath> <signature>/child::wnd[@title="Untitled - CompanyPad" and @class_name="CompanyPad"]/ child:: wnd[@class_name="Edit" and @ctrl_id=15]</signature> </wnd_xpath> </from> <to> <property_store_item id="test"> </property_store_item> </to> </data_transfer_item> </data_transfer_items> </data_transfer_action> </pre> <p>Note: The type of source location can be different from the destination location. For example, the source can be a field on an application screen while the destination can be an item in the account data bag.</p> <p>Specify the location of the source and destination for this action to be valid.</p> <p>You can also enable data transfer only when auto-logon is enabled.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Simulate keyboard input	<p>Sends the specified keyboard input to the destination application window.</p> <p>Example:</p> <p>To send Ctrl+Enter to the CompanyPad window:</p> <pre data-bbox="716 432 1458 716"><keyboard_input_action> <keyboard_inputs> <keyboard_input ctrl="1"><enter> </enter> </keyboard_input> </keyboard_inputs> <signature>/child::wnd[@title="Untitled - CompanyPad" and @class_name="CompanyPad"] / child::wnd[@class_name="Edit" and @ctrl_id=15]</signature> </keyboard_input_action></pre> <p>Note: You can define multiple sets of keyboard input. To add an input set, right-click on the keyboard_inputs data item and select Add Key.</p> <p>To make sure that the target window is on the foreground, specify whether to click the target window before sending the input.</p>
Click a menu option	<p>Clicks a specified menu item on the Windows application.</p> <p>Example:</p> <p>To click logon menu item on menu specify Signature of the window whose menu-item is clicked: /child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"] Menu Path: File/Login</p> <p>Note: You can enable clicking of a multi-level menu item by providing a separator '/' in the menu path. For example, File/Open.</p>
Notifies AccessAgent of application logoff	<p>Monitors and notifies whether a logoff from an application is successful.</p> <p>Example:</p> <p>To notify logoff is successful:</p> <pre data-bbox="716 1430 1458 1482"><notify_signed_out_action> </notify_signed_out_action></pre> <p>Tip: This action is useful when you configured multiple actions to occur upon logoff completion.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Change auto-fill policy	<p>Changes the auto-fill policy of an authentication service.</p> <p>Example:</p> <p>Change the auto-fill policy of auth1 to Auto-logout</p> <pre data-bbox="685 401 1154 638"><change_policy_action> <policy_value>auto_logout</policy_value> <auth_info> <direct_auth_info> <auth_id>auth1 </auth_id> </direct_auth_info> </auth_info> </change_policy_action></pre> <p>Note: The auto-fill policy can be one of the following auto_logout: Auto-fills the credentials without prompting the user, and performs auto-logout if only one set of user credentials is found. If several sets of credentials are found, the user is prompted to select the credentials to use.</p> <p>Always: Auto-fills the credentials if a single set of credentials is found, but does not log on automatically. If multiple credential sets are found, the user is prompted to select the credentials.</p> <p>Prompt for Re-login : Prompts the user upon logout from the application to log on again to the application. This prompt displays only if the auto-fill policy is set to always or automatic logon.</p> <p>Ask: Prompts the user to select the credentials to be auto-filled.</p> <p>Never: Disables auto-fill and auto-logout.</p> <p>Auto-logout: Auto-fills the credentials without prompting the user, and performs auto-logout only if one set of user credentials is found.</p>
Adds an entry to the audit log	<p>Enables audit logs for account data events that occurred.</p> <p>Example:</p> <p>To provide audit logs for successful application logon:</p> <pre data-bbox="685 1440 1357 1566"><acc_data_audit_log_action> <acc_data_bag_id>default_injection_bag</acc_data_bag_id> <event_code>1107296303</event_code> <return_code>0</return_code> </acc_data_audit_log_action></pre> <p>Note: You can enable audit logs for success or failure events upon application logon, and auto-fill, saving, or changing of credentials.</p>
Adds a custom entry to the audit log	Customizes audit logging.

Table 11. AccessStudio Actions (continued)

Action	Description
Show a dialog to capture logon credentials	<p>This action prompts a dialog box that requests for user logon information when capturing logon credentials for applications whose fields cannot be uniquely identified. For example, Owner-drawn Windows applications.</p> <p>Example:</p> <p>To prompt a dialog box for capturing user credentials:</p> <pre><observer_dlg_capture_action> <acc_data_bag id="default_capture_bag" </acc_data_bag> <save></save> </observer_dlg_capture_action></pre>
Show a dialog to capture change password credentials	<p>Prompts a dialog box that requests for change password information when capturing change password credentials for applications whose fields cannot be uniquely identified. For example, Owner-drawn Windows applications.</p> <p>Example:</p> <p>To prompt a dialog box for capturing change password credentials:</p> <pre><observer_dlg_change_password_action> <acc_data_bag id="default_capture_bag" </acc_data_bag> </observer_dlg_change_password_action></pre>
Start collecting keyboard input	<p>Starts collecting keyboard input from users for applications whose fields cannot be uniquely identified. For example, Owner-drawn Windows applications.</p> <p>Example:</p> <p>To start collecting keyboard input:</p> <pre><start_collecting_keyboard_input_action> <signature>/child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"]</signature> </start_collecting_keyboard_input_action></pre> <p>Note: This action is used with the Stop collecting keyboard input action.</p>
Stop collecting keyboard input	<p>Stops collecting keyboard input from the applications for which the Start collecting keyboard input action is run. These actions are typically used for applications whose fields cannot be uniquely identified. For example, an Owner-drawn Windows application.</p> <p>Example:</p> <p>To stop collecting keyboard input:</p> <pre><stop_collecting_keyboard_input_action><signature>/ child::wnd[@title="Form1"and@class_name="ThunderRT6FormDC"]</ signature></stop_collecting_keyboard_input_action></pre> <p>Note: This action is used with the Start collecting keyboard input action.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Perform SCR with the IMS Server	<p>Enables logon for an SCR-enabled Web application.</p> <p>The action initiates the password-less Session Challenge Response (SCR) authentication mechanism between AccessAgent and the Web application.</p> <p>Example:</p> <p>To enable SCR or password-less logon for a Web application that supports SCR, create a Perform SCR with the IMS server action inside a Web page completes loading trigger.</p> <p>Note: The site_info section of the sso_site containing this Perform SCR with the IMS server action must have an SCR-enabled application.</p> <p>The Web application on which the Perform SCR with the IMS server action is used must be SCR-enabled.</p> <p>This action has no effect on a Web application that is not configured for SCR.</p>
Register for SCR with the IMS Server	<p>Registers with an SCR-enabled Website.</p> <p>Example:</p> <p>To enable SCR or no-password registration for a Web application that supports SCR, create a Perform SCR with the IMS Server action after capturing user credentials for the application.</p> <p>Note: The contents in the account data bag is used to register with the specified Website.</p> <p>This action must be used with or after a Web page completes loading trigger. The credentials must be validated so that SCR registration succeeds, or it results in a failed SCR registration.</p>
Disable/enable an HTML element	<p>This action is used for enabling or not enabling an element on web pages.</p> <p>Note: The element must support IHTMLElement3 interface.</p>
Sets the visibility of an HTML element	<p>This action is used for showing or hiding an element on web pages.</p> <p>Note: The element must support IHTMLElement2 interface.</p>
Set a check-box state	<p>Sets the state of a check box on a Windows application by marking it or clearing it.</p> <p>Example:</p> <p>To clear a check box:</p> <pre data-bbox="686 1692 1414 1850"><wnd_set_check_box_state_action> <signature>/child::wnd[@title="Login" and @class_name="#32770"]/child::wnd[@class_name="Button" and @ctrl_id=1005] </signature> <set_checked>0</set_checked> </wnd_set_check_box_state_action></pre> <p>Note: There is an advanced option to specify to perform the action only if the auto-logon policy is enabled.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Executes a keyboard shortcut	<p>Sends a keyboard shortcut to a specified Windows application window.</p> <p>Example:</p> <p>Send the shortcut Ctrl+A to the CompanyPad window</p> <pre><wnd_shortcut_action> <signature>/child::wnd[@title="Untitled - CompanyPad" and @class_name="CompanyPad"]</signature> <shortcut_string ctrl="1">s</shortcut_string> </wnd_shortcut_action></pre> <p>Note: You can send shortcuts with several assistant keys, such as Ctrl+Alt+A.</p>
Close a window	<p>Closes a specified window.</p> <p>Example:</p> <p>Close the CompanyPad window:</p> <pre><wnd_close_window_action> <signature>/child::wnd[@title="Untitled - CompanyPad" and @class_name="CompanyPad"]</signature> </wnd_close_window_action</pre> <p>Note: Using the advanced options, you can specify to perform the action only if the automatic logon policy is available.</p>
Set the visibility of the window	Shows or hides a windows control.
Disable/enable a window	Enables or disables a windows control.
Disable/enable a menu item	Enables or disables a windows menu item.
Block user input	<p>Blocks all user input through the keyboard or mouse.</p> <p>This action might be used to prevent user input from interfering with credential injection.</p> <p>You can use the Unblock user input action to stop blocking user input.</p>
Unblock user input	<p>Resumes the user input through the keyboard or the mouse, which is previously blocked by Block user input action.</p> <p>This action does not take effect if preceded by a Block user input action.</p> <p>If multiple Block user input actions are in effect with non-expired timeout, this action reverts the effect of the latest action.</p> <p>If two Block user input actions are in effect, a single Unblock user input action leaves the system in a state where user input is blocked.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Kills a process	<p>Ends the specified process.</p> <p>Example:</p> <p>To end Internet Explorer:</p> <pre data-bbox="685 401 1179 478"><kill_process_action> <process_name>iexplore.exe</process_name> </kill_process_action></pre>
Run a VBScript or JScript	<p>Supports running custom scripts. The languages that are currently supported are VBScript and JScript.</p> <p>Example:</p> <p>To run a script to display current date and time:</p> <pre data-bbox="685 667 1284 800"><run_script_action> <script language="VBScript">document.write("Date : " & date()) document.write("Time : " & time())> </script> </run_script_action></pre> <p>Note: You can specify the script type in the Advanced Options section.</p>
Click a Java window	<p>Clicks a Java component.</p> <p>Note: The click is done by sending mouse input to the specified Java component.</p>
Start installing BHO for embedded IE browser	<p>Hooks into Windows applications, which host Internet Explorer Browser Controls to enable web page browsing. The hook enables listening to web page events and performing actions on the web page.</p> <p>Note: If no signature is specified, all windows that are created within the application with class name Internet Explorer_Server are hooked.</p> <p>The window must be an Internet Explorer_Server class.</p>
Stop installing BHO for embedded IE browser	<p>Stops hooking into Windows applications, which host Internet Explorer Browser Controls to enable web page browsing.</p> <p>Note: All the existing hooks remain until the hooked window is closed.</p>

Table 11. AccessStudio Actions (continued)

Action	Description
Click a window	<p>Clicks the specified window at the given position. The position must be defined in terms of coordinates corresponding to the upper left corner of that window.</p> <p>Example:</p> <p>Click the CompanyPad editor window at position 1,1 and do the click by sending user input:</p> <pre data-bbox="716 489 1458 751"><wnd_click_action> <signature>/child::wnd[@title="Untitled - CompanyPad" and @class_name="CompanyPad"]/ child::wnd[@class_name="Edit" and @ctrl_id=15]</signature> <position> <x>1</x> <y>1</y> </position> <do_simple_click>1</do_simple_click> </wnd_click_action></pre> <p>Note: You can specify to bring the window to the foreground before you click.</p> <p>There are two types of click simulations: Click by simulating the mouse actions (up or down), and click by providing input directly.</p>
Click a web element	<p>Clicks a specified element on a web page.</p> <p>Example:</p> <p>To enable clicking of an OK button of a Website, capture the signature of the OK button by using the Finder tool.</p> <p>Specify this action for a Web page completes loading trigger that fires when the page with the OK button is fully loaded.</p> <p>Note: Using this action, you can enable clicking of any Web element, such as a hyperlink, button, or image link.</p>
Wait for some time	Puts in a delay before running the actions that follows it.

Chapter 13. Frequently asked questions

See this topic for answers to common AccessStudio user questions.

What is Simple SSO Support?

Simple SSO Support is also known as a standard AccessProfile in AccessStudio.

See Chapter 3, “Standard AccessProfiles,” on page 17 for more information.

What is State Engine SSO Support?

State Engine SSO Support is also known as an Advanced AccessProfile. Advanced AccessProfiles are State Engine SSO Support in AccessStudio.

To create an Advanced AccessProfile, click the **New** icon from the AccessStudio toolbar and select **New Advanced AccessProfile**.

See Chapter 4, “Advanced AccessProfiles,” on page 51 for more information.

What is the difference between a standard AccessProfile and an advanced AccessProfile?

With a standard AccessProfile, you can configure support for applications that are based on commonly required information for applications.

You can use the AccessProfile Generator to create standard AccessProfiles. Standard AccessProfile support is sufficient for most applications.

Advanced AccessProfiles are useful for workflow automation.

What is the difference between an account data and an account data template?

Account data is the logon information required for authentication against an authentication service. Examples are user names and passwords. An account data template defines the nature of this information. For example, it defines that the user name field is not case-sensitive and is not a secret, while the password field is case-sensitive and is a secret.

Why do some of the AccessProfiles in the States diagram pane have red exclamation points beside them?

The red exclamation point beside an AccessProfile indicates that the XML for that AccessProfile is not structurally correct. This means that either some required elements or attributes are missing or some extra ones are added which AccessStudio does not recognize.

When you select the erroneous node in the States diagram pane, the Messages pane describes the errors that are associated with that state, trigger, or action, so that you can correct the problem. The red exclamation point disappears after the problem is resolved.

How would you define the relationship between a profile, an application, and an authentication service?

A profile can be written for only one application, but can correspond to multiple authentication services.

After I launch my application, I see a dialog box before the logon screen opens. How do I handle this application?

Generate an automated task (select **Other tasks** at the task selection step) for the dialog box.

How do I make the same AccessProfile work for multiple versions?

If the user interface remains the same, you can use the **Advanced settings** option in the AccessProfile Generator to remove variable information like version number.

If the user interface is different and the AccessProfile stops working, you can use the AccessProfile Generator to generate a new AccessProfile.

My screen title contains host/IP address of the server. How do I make the same AccessProfile work for multiple hosts?

Use the **Advanced settings** option in the AccessProfile Generator to remove variable information, like a host name or IP address.

How do I know whether there is already an existing AccessProfile written for an application?

Use the **File > Import data from IMS** option, which loads all available AccessProfiles in AccessStudio.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law :

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information; at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Privacy Policy Considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering uses other technologies that collect each user's user name, password or other personally identifiable information for purposes of session management, authentication, single sign-on configuration or other usage tracking or functional purposes. These technologies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at <http://www.ibm.com/privacy> and IBM’s Online Privacy Statement at <http://www.ibm.com/privacy/details/us/en> sections entitled “Cookies, Web Beacons and Other Technologies” and “Software Products and Software-as-a Service”.

Glossary

This glossary includes terms and definitions for IBM Security Access Manager for Enterprise Single Sign-On.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

account data

The logon information required to verify an authentication service. It can be the user name, password, and the authentication service which the logon information is stored.

account data bag

A data structure that holds user credentials in memory while single sign-on is performed on an application.

account data item

The user credentials required for logon.

account data item template

A template that defines the properties of an account data item.

account data template

A template that defines the format of account data to be stored for credentials captured using a specific AccessProfile.

action In profiling, an act that can be performed in response to a trigger. For example, automatic filling of user name and password details as soon as a sign-on window displays.

Active Directory (AD)

A hierarchical directory service that enables centralized, secure management

of an entire network, which is a central component of the Microsoft Windows platform.

Active Directory credential

The Active Directory user name and password.

Active Directory password synchronization

An IBM Security Access Manager for Enterprise Single Sign-On feature that synchronizes the ISAM ESSO password with the Active Directory password.

active radio frequency identification (active RFID)

A second authentication factor and presence detector. See also radio frequency identification.

active RFID

See active radio frequency identification.

AD See Active Directory.

administrator

A person responsible for administrative tasks such as access authorization and content management. Administrators can also grant levels of authority to users.

API See application programming interface.

application

A system that provides the user interface for reading or entering the authentication credentials.

application policy

A collection of policies and attributes governing access to applications.

application programming interface (API)

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

audit A process that logs the user, Administrator, and Helpdesk activities.

authentication factor

The device, biometrics, or secrets required as a credentials for validating digital identities. Examples of authentication

factors are passwords, smart card, RFID, biometrics, and one-time password tokens.

authentication service

A service that verifies the validity of an account; applications authenticate against their own user store or against a corporate directory.

authorization code

An alphanumeric code generated for administrative functions, such as password resets or two-factor authentication bypass.

auto-capture

A process that allows a system to collect and reuse user credentials for different applications. These credentials are captured when the user enters information for the first time, and then stored and secured for future use.

automatic sign-on

A feature where users can log on to the sign-on automation system and the system logs on the user to all other applications.

B

base distinguished name

A name that indicates the starting point for searches in the directory server.

base image

A template for a virtual desktop.

bidirectional language

A language that uses a script, such as Arabic and Hebrew, whose general flow of text proceeds horizontally from right to left, but numbers, English, and other left-to-right language text are written from left to right.

bind distinguished name

A name that specifies the credentials for the application server to use when connecting to a directory service. The distinguished name uniquely identifies an entry in a directory.

biometrics

The identification of a user based on a physical characteristic of the user, such as a fingerprint, iris, face, voice, or handwriting.

C

CA See certificate authority.

CAPI See cryptographic application programming interface.

Card Serial Number (CSN)

A unique data item that identifies a hybrid smart card. It has no relation to the certificates installed in the smart card

CCOW

See Clinical Context Object Workgroup.

cell

A group of managed processes that are federated to the same deployment manager and can include high-availability core groups.

certificate

In computer security, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority and is digitally signed by that authority. See also certificate authority.

certificate authority (CA)

A trusted third-party organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate. See also certificate.

CLI See command-line interface.

Clinical Context Object Workgroup (CCOW)

A vendor independent standard, for the interchange of information between clinical applications in the healthcare industry.

cluster

A group of application servers that collaborate for the purposes of workload balancing and failover.

command-line interface (CLI)

A computer interface in which the input and output are text based.

credential

Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a

user ID and password are credentials that allow access to network and system resources.

cryptographic application programming interface (CAPI)

An application programming interface that provides services to enable developers to secure applications using cryptography. It is a set of dynamically-linked libraries that provides an abstraction layer which isolates programmers from the code used to encrypt the data.

cryptographic service provider (CSP)

A feature of the i5/OS™ operating system that provides APIs. The CCA Cryptographic Service Provider enables a user to run functions on the 4758 Coprocessor.

CSN See Card Serial Number.

CSP See cryptographic service provider.

D

dashboard

An interface that integrates data from a variety of sources and provides a unified display of relevant and in-context information.

database server

A software program that uses a database manager to provide database services to other software programs or computers.

data source

The means by which an application accesses data from a database.

deployment manager

A server that manages and configures operations for a logical group or cell of other servers.

deployment manager profile

A WebSphere® Application Server runtime environment that manages operations for a logical group, or cell, of other servers.

deprovision

To remove a service or component. For example, to deprovision an account means to delete an account from a resource. See also provision.

desktop pool

A collection of virtual desktops of similar

configuration intended to be used by a designated group of users.

directory

A file that contains the names and controlling information for objects or other directories.

directory service

A directory of names, profile information, and machine addresses of every user and resource on the network. It manages user accounts and network permissions. When a user name is sent, it returns the attributes of that individual, which might include a telephone number, as well as an email address. Directory services use highly specialized databases that are typically hierarchical in design and provide fast lookups.

disaster recovery

The process of restoring a database, system, policies after a partial or complete site failure that was caused by a catastrophic event such as an earthquake or fire. Typically, disaster recovery requires a full backup at another location.

disaster recovery site

A secondary location for the production environment in case of a disaster.

distinguished name (DN)

The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute:value pairs, separated by commas. For example, CN=person name and C=country or region.

DLL See dynamic link library.

DN See distinguished name.

DNS See domain name server.

domain name server (DNS)

A server program that supplies name-to-address conversion by mapping domain names to IP addresses.

dynamic link library (DLL)

A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a DLL can be shared by several applications simultaneously.

E

enterprise directory

A directory of user accounts that define IBM Security Access Manager for Enterprise Single Sign-On users. It validates user credentials during sign-up and logon, if the password is synchronized with the enterprise directory password. An example of an enterprise directory is Active Directory.

enterprise single sign-on (ESSO)

A mechanism that allows users to log on to all applications deployed in the enterprise by entering a user ID and other credentials, such as a password.

ESSO See enterprise single sign-on.

event code

A code that represents a specific event that is tracked and logged into the audit log tables.

F

failover

An automatic operation that switches to a redundant or standby system or node in the event of a software, hardware, or network interruption.

fast user switching

A feature that allows users to switch between user accounts on a single workstation without quitting and logging out of applications.

Federal Information Processing Standard (FIPS)

A standard produced by the National Institute of Standards and Technology when national and international standards are nonexistent or inadequate to satisfy the U.S. government requirements.

FIPS See Federal Information Processing Standard.

fix pack

A cumulative collection of fixes that is released between scheduled refresh packs, manufacturing refreshes, or releases. A fix pack updates the system to a specific maintenance level.

FQDN

See fully qualified domain name.

fully qualified domain name (FQDN)

In Internet communications, the name of a host system that includes all of the subnames of the domain name. An example of a fully qualified domain name is rchland.vnet.ibm.com. See also host name.

G

GINA See graphical identification and authentication.

GPO See group policy object.

graphical identification and authentication (GINA)

A dynamic link library that provides a user interface that is tightly integrated with authentication factors and provides password resets and second factor bypass options.

group policy object (GPO)

A collection of group policy settings. Group policy objects are the documents created by the group policy snap-in. Group policy objects are stored at the domain level, and they affect users and computers contained in sites, domains, and organizational units.

H

HA See high availability.

high availability (HA)

The ability of IT services to withstand all outages and continue providing processing capability according to some predefined service level. Covered outages include both planned events, such as maintenance and backups, and unplanned events, such as software failures, hardware failures, power failures, and disasters.

host name

In Internet communication, the name given to a computer. The host name might be a fully qualified domain name such as mycomputer.city.company.com, or it might be a specific subname such as mycomputer. See also fully qualified domain name, IP address.

hot key

A key sequence used to shift operations

between different applications or between different functions of an application.

hybrid smart card

An ISO-7816 compliant smart card which contains a public key cryptography chip and an RFID chip. The cryptographic chip is accessible through contact interface. The RFID chip is accessible through contactless (RF) interface.

I

interactive graphical mode

A series of panels that prompts for information to complete the installation.

IP address

A unique address for a device or logical unit on a network that uses the Internet Protocol standard. See also host name.

J

Java Management Extensions (JMX)

A means of doing management of and through Java technology. JMX is a universal, open extension of the Java programming language for management that can be deployed across all industries, wherever management is needed.

Java runtime environment (JRE)

A subset of a Java developer kit that contains the core executable programs and files that constitute the standard Java platform. The JRE includes the Java virtual machine (JVM), core classes, and supporting files.

Java virtual machine (JVM)

A software implementation of a processor that runs compiled Java code (applets and applications).

JMX See Java Management Extensions.

JRE See Java runtime environment.

JVM See Java virtual machine.

K

keystore

In security, a file or a hardware cryptographic card where identities and private keys are stored, for authentication

and encryption purposes. Some keystores also contain trusted or public keys. See also truststore.

L

LDAP See Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol (LDAP)

An open protocol that uses TCP/IP to provide access to directories that support an X.500 model. An LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

lightweight mode

A Server AccessAgent mode. Running in lightweight mode reduces the memory footprint of AccessAgent on a Terminal or Citrix Server and improves the single sign-on startup duration.

linked clone

A copy of a virtual machine that shares virtual disks with the parent virtual machine in an ongoing manner.

load balancing

The monitoring of application servers and management of the workload on servers. If one server exceeds its workload, requests are forwarded to another server with more capacity.

lookup user

A user who is authenticated in the Enterprise Directory and searches for other users. IBM Security Access Manager for Enterprise Single Sign-On uses the lookup user to retrieve user attributes from the Active Directory or LDAP enterprise repository.

M

managed node

A node that is federated to a deployment manager and contains a node agent and can contain managed servers. See also node.

mobile authentication

An authentication factor which allows mobile users to sign-on securely to corporate resources from anywhere on the network.

N

network deployment

The deployment of an IMS™ Server on a WebSphere Application Server cluster.

node A logical group of managed servers. See also managed node.

node agent

An administrative agent that manages all application servers on a node and represents the node in the management cell.

O

one-time password (OTP)

A one-use password that is generated for an authentication event, and is sometimes communicated between the client and the server through a secure channel.

OTP See one-time password.

OTP token

A small, highly portable hardware device that the owner carries to authorize access to digital systems and physical assets, or both.

P

password aging

A security feature by which the superuser can specify how often users must change their passwords.

password complexity policy

A policy that specifies the minimum and maximum length of the password, the minimum number of numeric and alphabetic characters, and whether to allow mixed uppercase and lowercase characters.

personal identification number (PIN)

In Cryptographic Support, a unique number assigned by an organization to an individual and used as proof of identity. PINs are commonly assigned by financial institutions to their customers.

PIN See personal identification number.

pinnable state

A state from an AccessProfile widget that

can be combined to the main AccessProfile to reuse the AccessProfile widget function.

PKCS See Public Key Cryptography Standards.

policy template

A predefined policy form that helps users define a policy by providing the fixed policy elements that cannot be changed and the variable policy elements that can be changed.

portal A single, secure point of access to diverse information, applications, and people that can be customized and personalized.

presence detector

A device that, when fixed to a computer, detects when a person moves away from it. This device eliminates manually locking the computer upon leaving it for a short time.

primary authentication factor

The IBM Security Access Manager for Enterprise Single Sign-On password or directory server credentials.

private key

In computer security, the secret half of a cryptographic key pair that is used with a public key algorithm. The private key is known only to its owner. Private keys are typically used to digitally sign data and to decrypt data that has been encrypted with the corresponding public key.

provision

To provide, deploy, and track a service, component, application, or resource. See also deprovision.

provisioning API

An interface that allows IBM Security Access Manager for Enterprise Single Sign-On to integrate with user provisioning systems.

provisioning bridge

An automatic IMS Server credential distribution process with third party provisioning systems that uses API libraries with a SOAP connection.

provisioning system

A system that provides identity lifecycle management for application users in enterprises and manages their credentials.

Public Key Cryptography Standards (PKCS)

A set of industry-standard protocols used for secure information exchange on the Internet. Domino® Certificate Authority and Server Certificate Administration applications can accept certificates in PKCS format.

published application

An application installed on Citrix XenApp server that can be accessed from Citrix ICA Clients.

published desktop

A Citrix XenApp feature where users have remote access to a full Windows desktop from any device, anywhere, at any time.

R

radio frequency identification (RFID)

An automatic identification and data capture technology that identifies unique items and transmits data using radio waves. See also active radio frequency identification.

random password

An arbitrarily generated password used to increase authentication security between clients and servers.

RDP See remote desktop protocol.

registry

A repository that contains access and configuration information for users, systems, and software.

registry hive

In Windows systems, the structure of the data stored in the registry.

remote desktop protocol (RDP)

A protocol that facilitates remote display and input over network connections for Windows-based server applications. RDP supports different network topologies and multiple connections.

replication

The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target) and synchronizing the data in both locations.

revoke

To remove a privilege or an authority from an authorization identifier.

RFID See radio frequency identification.

root CA

See root certificate authority.

root certificate authority (root CA)

The certificate authority at the top of the hierarchy of authorities by which the identity of a certificate holder can be verified.

S

scope A reference to the applicability of a policy, at the system, user, or machine level.

secret question

A question whose answer is known only to the user. A secret question is used as a security feature to verify the identity of a user.

secure remote access

The solution that provides web browser-based single sign-on to all applications from outside the firewall.

Secure Sockets Layer (SSL)

A security protocol that provides communication privacy. With SSL, client/server applications can communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

Secure Sockets Layer virtual private network (SSL VPN)

A form of VPN that can be used with a standard web browser.

Security Token Service (STS)

A web service that is used for issuing and exchanging security tokens.

security trust service chain

A group of module instances that are configured for use together. Each module instance in the chain is called in turn to perform a specific function as part of the overall processing of a request.

serial ID service provider interface

A programmatic interface intended for integrating AccessAgent with third-party Serial ID devices used for two-factor authentication.

serial number

A unique number embedded in the IBM Security Access Manager for Enterprise Single Sign-On keys, which is unique to each key and cannot be changed.

server locator

A locator that groups a related set of web applications that require authentication by the same authentication service. In AccessStudio, server locators identify the authentication service with which an application screen is associated.

service provider interface (SPI)

An interface through which vendors can integrate any device with serial numbers with IBM Security Access Manager for Enterprise Single Sign-On and use the device as a second factor in AccessAgent.

signature

In profiling, unique identification information for any application, window, or field.

sign-on automation

A technology that works with application user interfaces to automate the sign-on process for users.

sign up

To request a resource.

silent mode

A method for installing or uninstalling a product component from the command line with no GUI display. When using silent mode, you specify the data required by the installation or uninstallation program directly on the command line or in a file (called an option file or response file).

Simple Mail Transfer Protocol (SMTP)

An Internet application protocol for transferring mail among users of the Internet.

single sign-on (SSO)

An authentication process in which a user can access more than one system or application by entering a single user ID and password.

smart card

An intelligent token that is embedded with an integrated circuit chip that provides memory capacity and computational capabilities.

smart card middleware

Software that acts as an interface between smart card applications and the smart card hardware. Typically the software consists of libraries that implement PKCS#11 and CAPI interfaces to smart cards.

SMTP See Simple Mail Transfer Protocol.

snapshot

A captured state, data, and hardware configuration of a running virtual machine.

SOAP A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet. See also web service.

SPI See service provider interface.

SSL See Secure Sockets Layer.

SSL VPN

See Secure Sockets Layer virtual private network.

SSO See single sign-on.

stand-alone deployment

A deployment where the IMS Server is deployed on an independent WebSphere Application Server profile.

stand-alone server

A fully operational server that is managed independently of all other servers, using its own administrative console.

strong authentication

A solution that uses multifactor authentication devices to prevent unauthorized access to confidential corporate information and IT networks, both inside and outside the corporate perimeter.

strong digital identity

An online persona that is difficult to impersonate, possibly secured by private keys on a smart card.

STS See Security Token Service.

system modal message

A system dialog box that is typically used to display important messages. When a system modal message is displayed,

nothing else can be selected on the screen until the message is closed.

T

terminal emulator

A program that allows a device such as a microcomputer or personal computer to enter and receive data from a computer system as if it were a particular type of attached terminal.

terminal type (tty)

A generic device driver for a text display. A tty typically performs input and output on a character-by-character basis.

thin client

A client that has little or no installed software but has access to software that is managed and delivered by network servers that are attached to it. A thin client is an alternative to a full-function client such as a workstation.

transparent screen lock

An feature that, when enabled, permits users to lock their desktop screens but still see the contents of their desktop.

trigger

In profiling, an event that causes transitions between states in a states engine, such as, the loading of a web page or the appearance of a window on the desktop.

trust service chain

A chain of modules that operate in different modes such as validate, map, and issue truststore.

truststore

In security, a storage object, either a file or a hardware cryptographic card, where public keys are stored in the form of trusted certificates, for authentication purposes in web transactions. In some applications, these trusted certificates are moved into the application keystore to be stored with the private keys. See also keystore.

tty See terminal type.

two-factor authentication

The use of two factors to authenticate a user. For example, the use of password and an RFID card to log on to AccessAgent.

U

uniform resource identifier

A compact string of characters for identifying an abstract or physical resource.

user credential

Information acquired during authentication that describes a user, group associations, or other security-related identity attributes, and that is used to perform services such as authorization, auditing, or delegation. For example, a user ID and password are credentials that allow access to network and system resources.

user deprovisioning

The process of removing a user account from IBM Security Access Manager for Enterprise Single Sign-On.

user provisioning

The process of signing up a user to use IBM Security Access Manager for Enterprise Single Sign-On.

V

VB See Visual Basic.

virtual appliance

A virtual machine image with a specific application purpose that is deployed to virtualization platforms.

virtual channel connector

A connector that is used in a terminal services environment. The virtual channel connector establishes a virtual communication channel to manage the remote sessions between the Client AccessAgent component and the Server AccessAgent.

virtual desktop

A user interface in a virtualized environment, stored on a remote server.

virtual desktop infrastructure

An infrastructure that consists of desktop operating systems hosted within virtual machines on a centralized server.

Virtual Member Manager (VMM)

A WebSphere Application Server component that provides applications with a secure facility to access basic

organizational entity data such as people, logon accounts, and security roles.

virtual private network (VPN)

An extension of a company intranet over the existing framework of either a public or private network. A VPN ensures that the data that is sent between the two endpoints of its connection remains secure.

Visual Basic (VB)

An event-driven programming language and integrated development environment (IDE) from Microsoft.

VMM See Virtual Member Manager.

VPN See virtual private network.

WS-Trust

A web services security specification that defines a framework for trust models to establish trust between web services.

W

wallet A secured data store of access credentials of a user and related information, which includes user IDs, passwords, certificates, encryption keys.

wallet caching

The process during single sign-on for an application whereby AccessAgent retrieves the logon credentials from the user credential wallet. The user credential wallet is downloaded on the user machine and stored securely on the IMS Server.

wallet manager

The IBM Security Access Manager for Enterprise Single Sign-On GUI component that lets users manage application credentials in the personal identity wallet.

web server

A software program that is capable of servicing Hypertext Transfer Protocol (HTTP) requests.

web service

A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available. See also SOAP.

Index

A

- AccessAgent, downloading
 - AccessProfiles 117
- accessibility vii
- AccessProfiles 75
 - advanced 51
 - applications that use Java applets 30
 - automation tasks 42
 - create 15
 - create logon for Java applications 28
 - download information 117
 - ID 8
 - log descriptions 114
 - Mainframe applications with HLLAPI support 38
 - Mainframe or cursor-based applications 33
 - other applications 40
 - playback logs 111
 - save information 118
 - terminal applications 31
 - test 113
 - TTY applications 31
 - upload information 118
 - version 8
 - Web applications 23
 - Windows applications 17
- AccessStudio
 - about 1
 - actions 152
 - advanced concepts 5
 - basic concepts 3
 - features and benefits 1
 - frequently asked questions 163
 - icons 8
 - interface 8
 - signatures 75
 - executable 78
 - HTML 82
 - Java windows 85
 - operators 77
 - significance of the root node ('/') 78
 - supported axes 76
 - supported types 77
 - web pages 80
 - windows 79
 - signatures for frames 83
 - standard AccessProfiles 17
 - system architecture 2
 - triggers 137
- AccessStudio interface
 - data type pane 10
 - details pane 10
 - menu bar 9
 - message pane 15
 - tool bar 9
- account data item templates 131
 - about 131
 - view 133

- account data templates
 - about 131
 - view 133
- advanced AccessProfiles 51, 69
 - add random password fields 70
 - application objects 56
 - automatic logon 54
 - check and prompt for relogin 69
 - create 51
 - creating samples
 - password change 63
 - custom requirements 71, 72, 74
 - edit 53
 - logon workflow 62
 - signature captures 56
 - state engines 57, 61
- application objects
 - create 127
 - manage 127
 - modify 129
- authentication services 119
 - associate with AccessProfiles 119
 - create 121
 - direct auth-info 119
 - group and group links 125
 - groups and group links 124
 - indirect auth-info 120
 - modify 124
 - modify group 126
 - modify group links 126
- automation tasks 41
 - advanced options 50
 - click controls 43
 - close screens 47
 - create 49
 - custom audit logs 48
 - delete 49
 - disable controls 44
 - edit 49
 - enable controls 44
 - enter text 45
 - hide controls 45
 - pause actions 45
 - press keys 46
 - run plug-ins 46
 - select menu items 47

E

- education vii

F

- functions, validation 87

G

- glossary 169

H

- HLLAPI 73

I

- IBM
 - Software Support vii
 - Support Assistant vii
- IMS Server
 - AccessStudio integration 2
 - authentication service records 120
 - data backup 135
 - download AccessProfiles 117
 - menu bar functions 9
 - upload information 118

L

- log level 111

M

- Mainframe or cursor-based applications
 - change password AccessProfiles 35
 - create logoff AccessProfiles 37
 - logon AccessProfiles 34

O

- Observer log 111

P

- Play back logs 111
- plug-in APIs
 - about 87
 - customized actions 88
 - customized triggers 87
 - specifications 88
 - ListControl 108
 - observer script debug object 101
 - property container class 91
 - property manager 93
 - runtime object 88
 - user data provider class 96
 - window controller class 101
- problem-determination vii
- publications
 - statement of good security practices vii

S

- Script name 72
- signatures
 - about 75
 - executable 78
 - frame-related 83

- signatures (*continued*)
 - HTML 82
 - Java windows 85
 - operators 77
 - significance of the root node ('/') 78
 - supported axes 76
 - supported types 77
 - web pages 80
 - windows 79
- standard AccessProfiles
 - create 17
 - edit 48

T

- terminal applications
 - change password AccessProfiles 32
 - logon AccessProfiles 31
- training vii
- triggers 71, 137

W

- Web applications
 - change password AccessProfiles 25
 - logoff AccessProfiles 26
 - logon AccessProfiles 24
 - other task AccessProfiles 27
- Windows applications
 - change password AccessProfiles 20
 - logoff AccessProfiles 21
 - logon AccessProfiles 18
 - other task AccessProfiles 22



Printed in USA